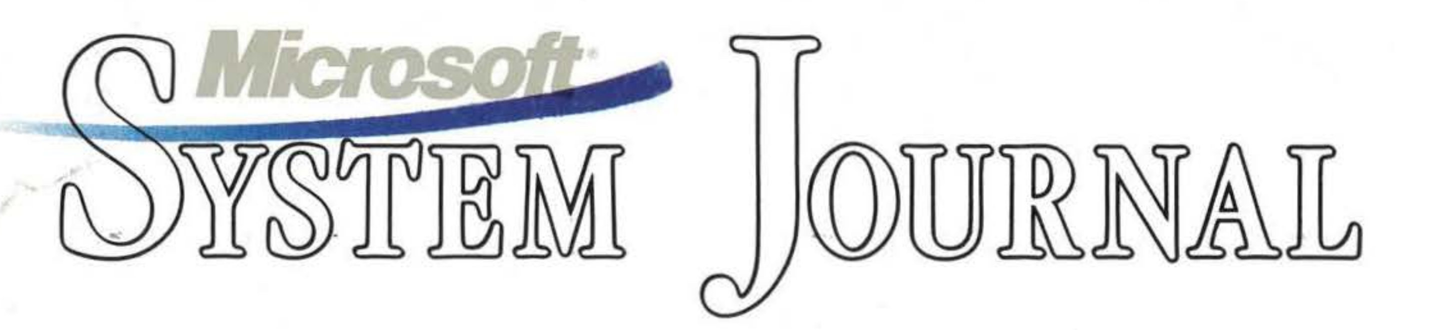


ISSN 0933-9434 | Juli/August 1988 | 2. Jg./Heft 4

ÖS 150,- | DM 19,80 | sfr 19,80

OS/2-Wissen verständlich:

Die Grafikprogrammierschnittstelle des Presentation Managers

Koordination von Threads mit Semaphoren

Das Dateisystem von MS-OS/2

Gerätetreiber unter MS-DOS

Programmierbeispiele für Einsteiger:

Dynamische Speicherverwaltungstechniken in C

Terminalemulation mit QuickBASIC

TIFF: Das neue Standardformat für Grafiken

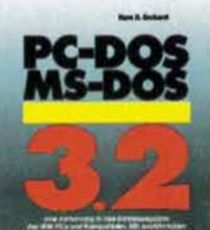
Kompaktes Wissen über Betriebs-systeme



H. H. Gerhardt: **DOS 3.3 für PCs und Personal System/2**
1988, 334 Seiten
Eine leichtverständliche Einführung mit Beispielen, Übungen und alphabetischer Befehlsübersicht. Für Anfänger und Fortgeschrittene. Mit umfangreichem Anhang: Begriffserklärung, Zeichencode-Tabellen, Kursdarstellung zum Thema »Netzwerk«, Fehlerübersicht, Übersicht zu den »DOS Function Calls« und »DOS Interrupt Calls«.
Bestell-Nr. 90547 ISBN 3-89090-547-1
DM 69,–/sFr 63,50/öS 538,20

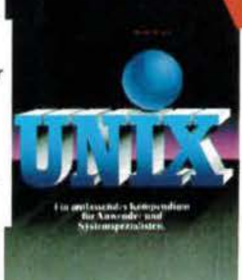
NEU

P. Monadjemi: **PC-Programmierung in Maschinensprache**
1988, ca. 300 Seiten, inkl. Diskette
Eine Einführung in die 8086/8088-Maschinenspracheprogrammierung mit vielen Beispielen. Zahlreiche Assemblerroutinen und ein ASCII-Texteditor sind auf der beigefügten Diskette enthalten. Mit allen Befehlen und einem Assembler-Glossar im Anhang.
Bestell-Nr. 90503 ISBN 3-89090-503-X
DM 69,–/sFr 63,50/öS 538,20



H. H. Gerhard: **PC-DOS/MS-DOS 3.2**
1987, 299 Seiten, inkl. Diskette
Eine Einführung in die wichtigsten Grundlagen zur Bedienung Ihres PCs mit DOS sowie ein hilfreiches Nachschlagewerk für jeden DOS-Anwender. Mit ausführlichen Befehlsbeschreibungen, alphabetischem Nachschlageteil und vielen Beispielen im PC-DOS- und MS-DOS-Format auf Diskette.
Bestell-Nr. 90519 ISBN 3-89090-519-6
DM 59,–/sFr 54,30/öS 460,20

NEU



H. Drees: **Unix**
1988, ca. 600 Seiten
Ein umfassendes Kompendium für Anwender und Systemspezialisten. Es beschreibt kein spezielles Unix, sondern Unix schlechthin und stützt sich dabei auf den Leistungsstandard, der durch Unix V repräsentiert wird. Mit vielen Beispielen und Anregungen.
Bestell-Nr. 90494 ISBN 3-89090-494-7
DM 89,–/sFr 81,90/öS 694,20



U. Schmidt
Daten retten mit Norton Utilities
2., überarb. Aufl., 1987, 301 Seiten
Das Übungsbuch zum Programmpaket mit ausführlichen Beispielen zur Rettung gelöschter Daten, Disketten/Festplatten-Organisation, universellen Textsuche und vielem mehr.
Bestell-Nr. 90505
ISBN-3-89090-505-6
DM 49,–/sFr 45,10/öS 382,20

Dr. Norbert Meder: **MS-OS/2**
1988, 304 Seiten
Einführung und Überblick über die neuen Programmiermöglichkeiten von MS-OS/2. Den Schwerpunkt bildet der neue Befehlsvorrat von MS-OS/2. Die einzelnen Befehle werden anhand von Beispielen leichtverständlich erläutert. Auch die Batch-Programmierung wird behandelt.
Bestell-Nr. 90512
ISBN 3-89090-512-9
DM 79,–/sFr 72,20/öS 616,20

U. Schmidt: **Das MS-Windows-Kompendium**
1988, ca. 350 Seiten, inkl. zwei Disketten
Tips, Tricks und Training, eine ausführliche Programmdokumentation und Hilfen bei der Anpassung verschiedener Softwarepakete finden Sie in diesem umfangreichen Kompendium. Für Windows 2.0 und 386. Auf den zwei Disketten finden Sie ein ausführliches Hilfsprogramm mit Installationsprogramm für die Festplatte, Utilities und zahlreiche Beispiele.
Bestell-Nr. 90558 ISBN 3-89090-558-7
DM 69,–/sFr 63,50/öS 538,20

NEU

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

Irrtümer und Änderungen vorbehalten.



Zeitschriften · Bücher
Software · Schulung

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0.
SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656,
ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0,
Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526
Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

Fragen Sie bei Ihrem Buchhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!


189/805

Ausgabe Juli/August 1988





Microsoft Windows

MS-OS/2 & Presentation Manager

QuickBASIC

Microsoft C und QuickC

Assembler

Rubriken

Programmentwicklung für Windows:

Ein Interview mit den Excel-Entwicklern

Bei der Umsetzung von Microsoft Excel in die PC-Umgebung bedurfte es größerer Anstrengungen, als bei einer einfachen Portierung der Macintosh-Version. Das Projekt-Team wollte eine Tabellenkalkulation entwickeln, die ein vollkommen geräteunabhängiges, grafikorientiertes Produkt ist und nicht nur das Aussehen und die Bedienung der Macintosh-Version haben, sondern auch wesentliche Verbesserungen in punkto Geschwindigkeit und Funktionalität bieten sollte. Um diese Ziele zu erreichen, wurde die Kalkulation für die Windows-Umgebung komplett neu geschrieben.

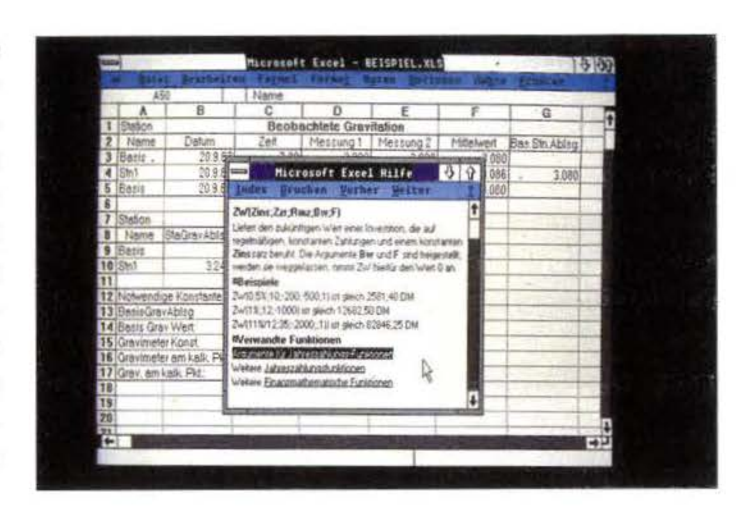


Bild 1: *Das Hilfe-Menü bietet den Benutzern detaillierte Unterstützung, zum Beispiel ein umfangreiches Glossar für alle Kalkulationsbegriffe, ein komplettes Online-Referenzhandbuch (wie es hier zu sehen ist), Tastaturübersichten und ein interaktives Online-Lernprogramm.*

Microsoft System Journal sprach mit dem Entwicklungsteam, um mehr über die Planung, die Entwicklung und ihre gemeinsamen Anstrengungen zu erfahren, die sie bei dieser komplexen Arbeit der Entwicklung von Microsoft Excel aufwenden mußten.

MSJ: *Was waren die eigentlichen Ziele beim Microsoft-Excel-Projekt? Wie wurden sie formuliert und in welcher Weise waren sie einzigartig für dieses Projekt?*

Das eigentliche Ziel war die Portierung der Macintosh-Version von Microsoft Excel nach Windows. Wir sahen, daß wir auf dem Macintosh ein fantastisches Produkt hatten und wollten es unter einer ähnlichen grafischen Benutzerschnittstelle auf dem PC zum Laufen zu bringen.

Als Teil der Formulierung der Projektziele versuchten wir, die besonderen Anforderungen der PC-Benutzer zu verstehen. Wir achteten vor allem auf Fähigkeiten, die in vorhandenen Tabellenkalkulationen wie Lotus 1-2-3 vorhanden sind und wollten sicherstellen, daß unsere Software mindestens kompatibel bzw. konsistent mit diesen Fähigkeiten wird. Dann fügten wird die weiteren Fähigkeiten hinzu, die die Mac-Version zu einem so großen Erfolg gemacht haben.

MSJ: *Microsoft Excel wurde für Microsoft Windows entwickelt. In welchen Bereichen war Windows dem Projekt förderlich, in welchen hinderlich?*

Windows war dem Projekt ganz eindeutig förderlich, da es die grafische Umgebung bietet, die Microsoft Excel benötigt. Windows ist eine vollständige, Bitmap-orientierte und sehr gut unterstützte Umgebung, die geräteunabhängige Treiber für all die verschiedenen Geräte beinhaltet, die in der PC-Welt existieren. Aus dieser Sicht betrachtet hätten wir Microsoft Excel ohne ein Produkt wie Windows gar nicht entwickeln können: Der erforderliche Mehraufwand, der für eine so leistungsfähige Umgebung wie Windows erforderlich ist, hätte das Projekt unmöglich gemacht.

Windows war insofern hinderlich, als es eine sehr komplexe Umgebung ist. Teilweise liegt das daran, daß die PC-Welt selbst so komplex ist - die Anzahl der Geräte, die unterstützt werden müssen und die Probleme, die sich durch die Geräteunabhängigkeit ergeben, machen die PC-Welt schwieriger als die Apple-Welt.

MSJ: *Microsoft Excel läuft unter Windows 2.0, Windows/386 und bald auch unter OS/2. Was mußte berücksichtigt werden, um die Langlebigkeit und Portabilität des Programms sicherzustellen, und wurde die Aufgabe dadurch schwieriger?*

Es ist einfach so, daß die Macintosh-Umgebung sich von der Windows-Umgebung unterscheidet und diese wieder anders ist, als die Presentation-Manager-Umgebung. Häufig erkennt man diese Unterschiede erst, wenn man wirklich damit zu tun hat, wie das bei diesem Projekt der Fall war. Wir haben bei der Umsetzung vom Macintosh nach Windows eine Menge gelernt, was das Verständnis von Microsoft Excel angeht, und wie wir es noch umgebungsunabhängiger machen können. Wir haben Dinge gelernt, die wir niemals erwartet haben, als wir die Portierung nach Windows begannen.

Wir verwenden bei der Portierung von einer Umgebung in eine andere ein Konzept, daß wir »Kerncode« nennen. Das bedeutet, es wird so viel wie möglich herausgenommen, was nicht umgebungsspezifisch ist. Dann nimmt man den umgebungsabhängigen Code und versucht, das zu isolieren, was in den Umgebungen verschieden ist und stellt fest, wie der Code so angepaßt werden kann, daß er von beiden verwendet werden kann.

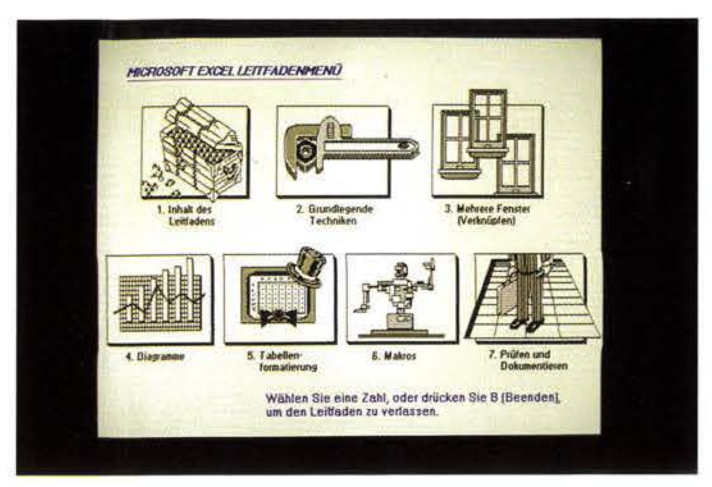


Bild 2: *Microsoft Excel bietet einen umfangreichen Satz an Tools, die dazu verwendet werden können, ausgefeilte Kalkulationsanwendungen und Berichte mit professioneller Qualität zu erstellen.*

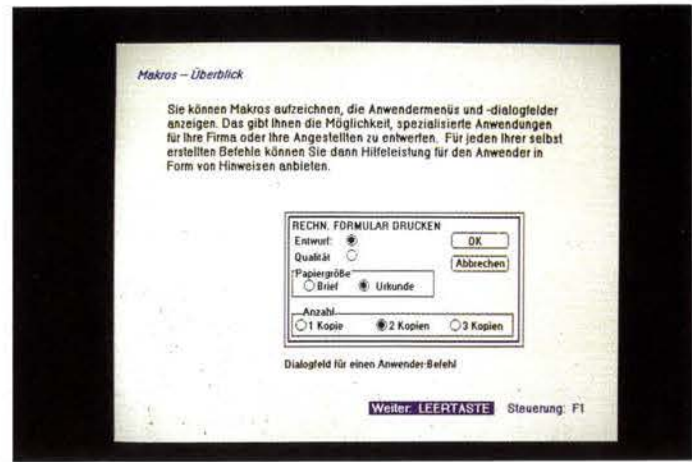


Bild 3: *Die Makros automatisieren nicht nur wiederholte Arbeitsschritte (zum Beispiel mit dem Makro-Rekorder), sondern ermöglichen auch die extensive eigene Gestaltung des Bildschirmaufbaus.*

Microsoft Excel wird bald auch an den Presentation Manager unter MS-OS/2 angepaßt werden. Der Presentation Manager unterscheidet sich ein wenig von Windows. Microsoft Excel wird so geändert werden müssen, daß diese Unterschiede unterstützt werden, aber es wird sicherlich problemlos laufen, wenn das erfolgt ist. Wir glauben, das nach der Implementierung in zwei Umgebungen die dritte wesentlich einfacher sein wird.

MSJ: *Einige von Ihnen waren an der Entwicklung von Microsoft Excel für den Macintosh beteiligt. Was war der größte Unterschied des PC-Projekts und wie hat Ihnen die Mac-Erfahrung bei diesem Projekt genutzt?*

Der interessanteste Aspekt des PC-Projekts war die Änderung der Speicherverwaltung. Ein Grund dafür war, daß wir eine bessere Adressierbarkeit haben wollten, als wir sie auf dem Macintosh haben. Wichtiger jedoch war, wie man die bessere Adressierbarkeit in der PC-Welt realisieren kann - zur Zeit wird das in der Regel mit EMS-Karten durchgeführt. Gewöhnlich haben die Karten ein Speicherfenster von 64 Kbyte und es können sich in einem Speicherfenster vier Seiten befinden. Aus Effizienzgründen sollte man die Seiten kleiner als 64 Kbyte halten. Meistens verwendet man eine Größe von etwa 16 Kbyte, damit man so viele wie möglich davon haben kann. Der ganze Entwurf der Datenstrukturen mußte geändert werden, um diese »Brocken« zu unterstützen.

MSJ: *Wie wurde das Microsoft-Excel-Projekt in einzelne Aufgaben aufgeteilt?*

Es wurde in einzelne Aufgaben aufgeteilt, indem Microsoft Excel für den Macintosh sorgfältig untersucht und ein Plan gemacht wurde, wie der existierende Code der Macintosh-Version nach Windows portiert werden könnte.

Es waren eine ganze Reihe von erfahrenen Leuten daran beteiligt - Leute, die schon an der Macintosh-Version mitgearbeitet hatten und die eine Menge von grafischen Benutzerschnittstellen und Umgebungen verstehen. Im wesentlichen verteilten wir die Bereiche nach der Erfahrung, die die Leute hatten. Die Diagramme wurden zum Beispiel jemand gegeben, der genau versteht, wie das in der Mac-Umgebung funktioniert und der bei den Änderungen für Windows sogar ein noch größerer Experte wurde.

MSJ: *Was ist Ihrer Meinung nach die ideale Größe für ein Projekt-Team?*

Die Teamgröße lag zwischen 8 und zehn Leuten. Irgendwann über zehn Leuten wird ein Projekt zu groß: Es entwickeln sich Kommunikationsprobleme und der Durchsatz wird geringer. Je mehr Leute damit befaßt sind, desto mehr Zeit bringt man damit zu, sie über Änderungen zu informieren und sicherzustellen, daß jeder versteht, was die anderen machen. Wir ziehen es vor, Projekt-Teams unterhalb dieser Zehn-Leute-Grenze zu haben. Wenn wir das nicht können, verwenden wir separate Unterprojekte und erstellen sehr gute, sauber definierte Schnittstellen zwischen den Untergruppen, so daß es nicht erforderlich ist, daß sie stark miteinander zu tun haben.

MSJ: *Erzählen Sie uns etwas über die Geheimnisse und Strategien, die sicherstellten, daß das Projekt gelang.*

Unsere Strategie besteht darin, daß wir den Einzelnen viel Verantwortung übertragen, so daß sie sich persönlich an dem beteiligt fühlen, was sie tun und daß sie wirklich stolz auf das sind, was sie geschaffen haben. Wir machen hier keine Vorschriften, die regeln, wie die Leute Probleme zu lösen haben und bestehen nicht auf Entwürfen, die sie zu implementieren haben.

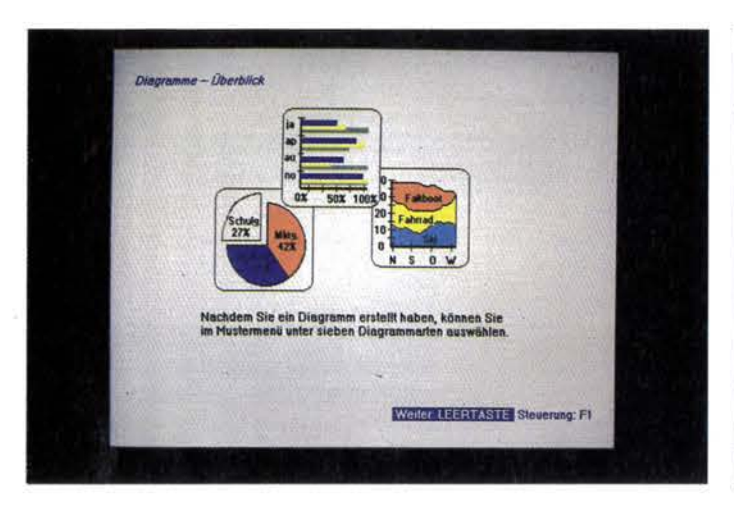


Bild 4: *Die insgesamt 44 verschiedenen Grafiktypen bieten eine große Flexibilität. Diagramme können nach individuellen Erfordernissen modifiziert und für Vergleiche auch neben- und übereinander gelegt werden.*

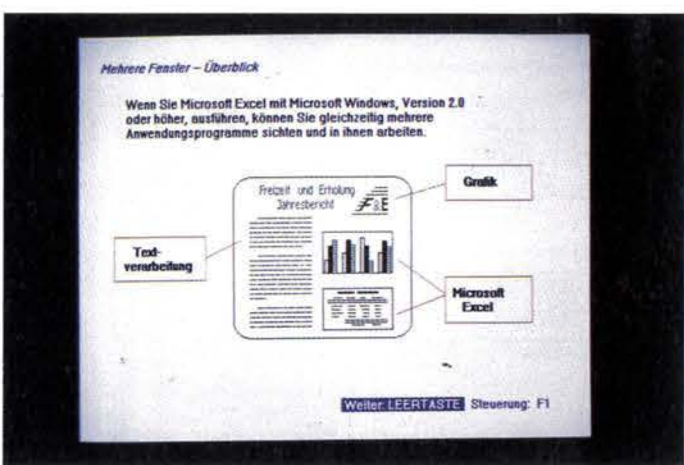


Bild 5: *Die Windows-Umgebung erhöht die Vielseitigkeit von Microsoft Excel, da der Benutzer Grafiken und Tabellen aus Microsoft Excel in andere Anwendungen (z.B. Write) integrieren kann.*

Es ist mehr so, daß die Leute zu Experten in Ihrem Bereich werden und dann selbst die Initiative übernehmen. Sie sind auch dafür verantwortlich, daß eine Lösung gefunden wird und daß alles innerhalb vernünftiger Zeiten geschieht.

Ein wichtiger Teil davon ist, daß die Team-Mitglieder am Entwurf und der Planung des Projekts teilhaben. Sie übernehmen die Verantwortung für das, was sie tun und wie lange sie dafür brauchen; es ist ein Bottom-Up-Entwurf. Jemand analysiert sorgfältig die notwendige Arbeit und verpflichtet sich dann, diese Arbeit durchzuziehen.

MSJ: *Ganz offensichtlich haben Sie dafür gesorgt, daß Anwender von Lotus 1-2-3 einen leichten Übergang nach Microsoft Excel haben. Haben Sie andere Zugeständnisse machen müssen, um Lotus-Benutzern entgegenzukommen?*

Wir haben eine sorgfältige Analyse der Fähigkeiten von Lotus 1-2-3 Version 2 durchgeführt. Wir haben dafür gesorgt, daß die Funktionalität komplett vorhanden ist, so daß Lotus-Benutzer nicht glauben, irgendetwas aufgeben zu müssen, das für sie nützlich ist. Die Entwicklung des Makro-Umsetzers war eine sehr umfangreiche Aufgabe. Die Makrosprache von Microsoft Excel arbeitet total anders als die 1-2-3-Makros, denn wir verwenden nicht Tasten als Grundlage. Wir haben die Lotus-Schnittstelle auch nicht direkt umgesetzt, wie das andere Produkte machen. Wir wollten ein intelligentes Programm haben, das die Makros analysieren und feststellen kann, was die Absicht des Lotus-Makros ist und es dann in die gleiche Art Makros für Microsoft Excel umsetzt.

Wenn es den 1-2-3-Standard nicht gegeben hätte, hätten wir einige Dinge sicherlich anders gelöst - auf Arten, die wir für effizienter und gelungener halten. Lotus 1-2-3 ist nun mal in den USA ein Standard.

Da sehr viele Leute 1-2-3 kennen und damit umgehen können, mußten wir uns irgendwie an diesen Standard halten. Wir konnten also nicht alle Fähigkeiten so implementieren, wie wir es gerne getan hätten.

MSJ: *Wie wichtig war bei diesem Projekt die Devise »Mach' es schneller«? Wie haben Sie festgestellt, daß das Programm schnell genug ist?*

Die Geschwindigkeit ist bei Tabellenkalkulationen sehr wichtig. Das haben wir von Microsoft Excel für den Macintosh gelernt haben; sie ist einer der Gründe dafür, daß das Programm so gut angekommen ist. Wir haben uns sehr stark auf die Geschwindigkeit konzentriert und die Benutzer meinten, daß das Programm das ist, was wir »knackig« nennen. Es reagiert schnell auf das, was sie tun wollen.

Deshalb haben wir die Geschwindigkeit bei Microsoft Excel zur Aufgabe mit höchster Priorität gemacht. Das führte dazu, daß wir sehr ausführliche Benchmarks mit vorhandenen Programmen und Microsoft Excel für MS-DOS angestellt haben, während wir es erstellten. Jedesmal, wenn wir eine Testversion fertig hatten, haben wir alle Benchmarks wieder laufenlassen. Wenn irgendetwas langsamer wurde, haben wir schnell herausbekommen, woran es lag. Mit den Benchmarks haben wir uns Ziele gesetzt; die regelmäßige Verwendung ermöglichte es uns, diese Ziele zu erreichen. Wir verwendeten so viel wie möglich bestehende Entwürfe, um auch noch das letzte Bißchen Geschwindigkeit herauszuquetschen, ohne irgendetwas wegzuwerfen. Dies macht man, indem man Benchmarks durchführt, die vermutlich langsamen Teile herausfindet und dann eine detaillierte Analyse mit Profilern durchführt. Durch diese Analyse kann man herausfinden, was langsam ist. Man kann dann entscheiden, ob man einen begrenzten Neuentwurf macht oder nur besseren Code für diesen Bereich schreibt.

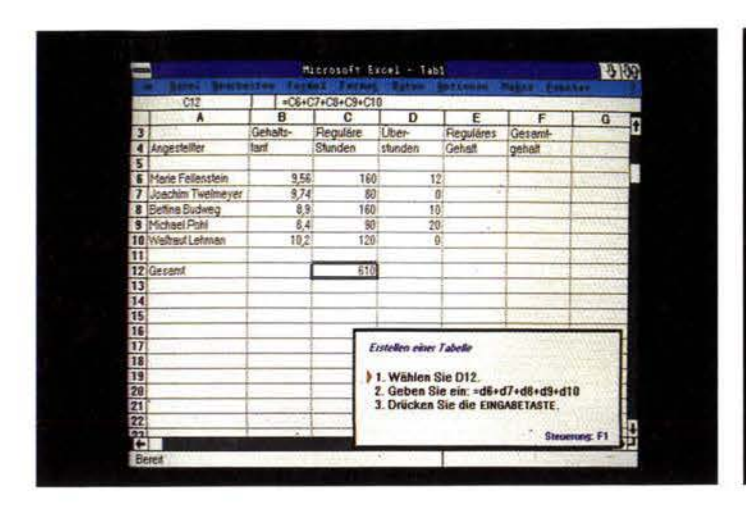


Bild 6: *Microsoft Excel kann mit dem interaktiven Lernprogramm leicht erlernt werden. Es analysiert die Benutzereingaben und stellt sicher, daß die richtigen Arbeitsabläufe befolgt und die korrekten Antworten gegeben werden.*

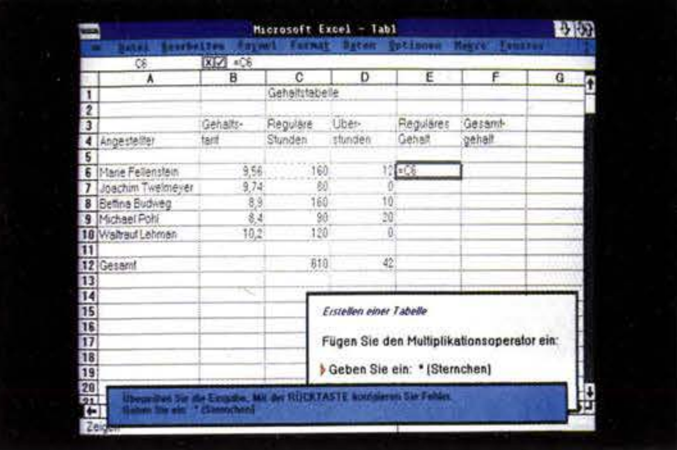


Bild 7: *Ein Fehler des Benutzers führt dazu, daß das Lernprogramm Tips gibt und so dem Benutzer die Gelegenheit gibt, die Antwort noch einmal zu überdenken und eine neue Antwort einzugeben.*

In einer grafischen Umgebung sind die Bereiche, die langsam sind, nicht die, die die Neuberechnung durchführen. Was langsam ist, ist die Bildschirmausgabe - ganz einfach die Bewegung all der vielen Bits. Wir haben erhebliche Zeit damit zugebracht, zu verstehen, wo die Zeit bei der Bildschirmanzeige verschwendet wird. Wie sich zeigte, wird ein hoher Prozentsatz der Zeit beim Scrollen damit verbracht, die Bits auf dem Bildschirm von einer Stelle zur anderen zu verschieben. Der Grund dafür liegt darin, daß die EGA-Karte, die unter anderem von uns unterstützt wird, eine Menge Wait-States benötigt. Es wird eine Menge Zeit dafür benötigt, eine Bit-Spalte zu verschieben, ganz einfach weil die EGA-Karte so langsam ist. Wir kennen die Karte nun genau und haben viel Zeit darauf verwendet, die Bildschirmausgabe so schnell wie möglich zu machen.

MSJ: *Unter anderen Neuheiten bietet Microsoft Excel auch das Windows-DDE-Protokoll (Dynamic Data Exchange). Wie schwierig war dieser Teil des Projekts und was haben Sie dabei über die Implementierung von DDE gelernt?*

Im Prinzip war DDE sehr einfach. Wir machten einen Entwurf, beschrieben, was wir dafür benötigten und schafften es, mit einer Untermenge von neun Befehlen auszukommen, die alles machen, was wir brauchen.

Andererseits war die Implementierung sehr schwierig. Da DDE ein asynchrones Kommunikationsprotokoll ist, muß das Produkt Meldungen jederzeit bearbeiten können, ganz gleich, was das Produkt gerade macht. Das macht es so komplex. Microsoft Excel war ursprünglich nicht so konzipiert, daß es Meldungen zu jeder beliebigen Zeit bearbeiten kann. Wir mußten genau analysieren, wie wir diese Fähigkeit implementieren konnten.

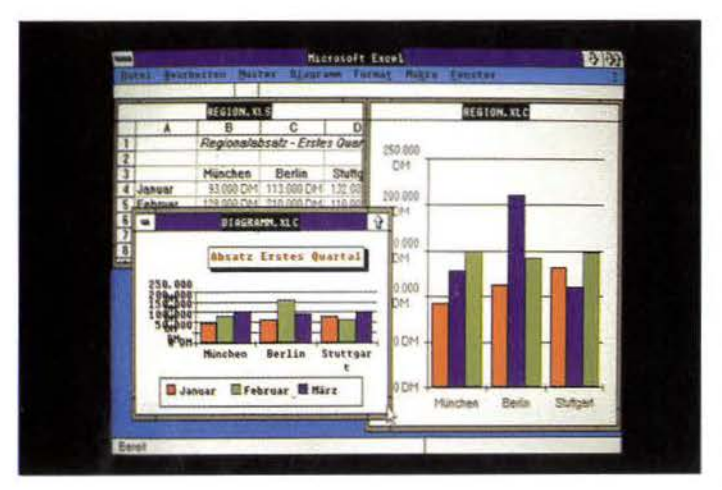
Wir mußten Entwurfsänderungen vornehmen, um Meldungen in eine Warteschlange zu stellen oder um sie zu bearbeiten, während das Programm sich in der Makrosprache befindet, gerade eine Neuberechnung durchführt oder andere zeitintensive Arbeiten durchführt. Die Implementierung dauerte wesentlich länger als wir erwartet hatten, doch rückblickend muß ich sagen, daß das daran lag, daß wir nicht von vorneherein alle Dinge berücksichtigten, die zur Unterstützung dieses Protokolls notwendig sind.

MSJ: *Das Microsoft-Excel-Projekt benötigte drei Jahre. Wie wichtig war die Stimmung der Gruppe in dieser Zeit?*

Sehr wichtig. Man kann keine derart großen und komplexen Projekte durchziehen, ohne daß die Leute motiviert sind und sich in gewissem Sinne als Schöpfer fühlen. Wir haben bei all unseren Projekten immer eine sehr gute Stimmung gehabt, weil die Leute fühlen, daß es ihre Arbeit ist, die da gemacht wird.

Teilweise beruht die gute Stimmung darauf, daß die Leute, die den Code schreiben, auch verstehen, um welche Features es geht. Die Features werden nicht von jemand vorgeschrieben der Spezifikationen schreibt und der sagt: »Und jetzt führen wir das aus.« Die Entwicklungsgruppe ist aktiv an der Analyse der Features beteiligt. Sie versuchen zu verstehen, was wir erreichen wollen, machen Vorschläge und sagen, was sie für gut halten und was nicht.

Jemand der die falsche Arbeit macht, weiß das in der Regel. Es ist sehr selten, daß jemand für sechs Monate in der falschen Position gewesen ist und man dann zu der Person sagt: »Das ist scheußlich.« Stattdessen wird auf dem ganzen Weg eine leitende Hand geboten. Das hilft den Qualm zu vertreiben, wenn es einmal zu viel geworden ist: zeigen, was wichtig ist, vielleicht auf eine bessere Lösung hinweisen oder auch empfehlen, einen Schritt zurückzugehen und darüber nachzudenken, was bisher gemacht wurde, bevor weitergemacht wird.



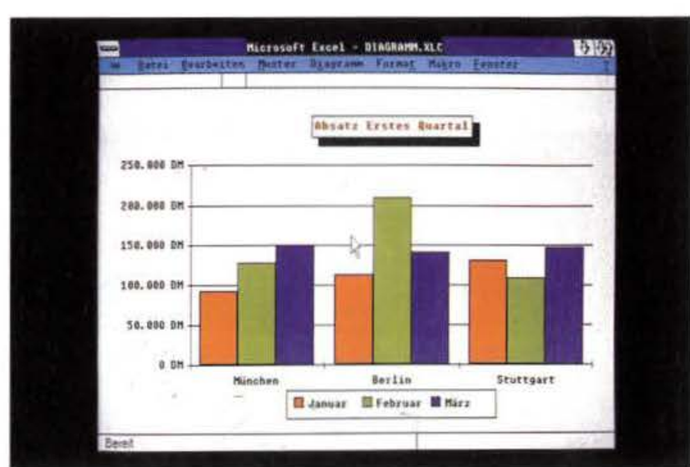


Bild 8: *Selbst wenn Excel nicht in der Windows-Umgebung läuft, bietet es umfangreiche Fenster-Möglichkeiten. Hier sind zwei Diagramme und die zugrundeliegenden Tabellen zu sehen, die verschiedene Ansichten derselben Daten zeigen.*

Bild 9: *Die Diagramme können vom Benutzer in vielen Einzelheiten durch selbstgewählte Farben, Ränder, Schattierungen, Schriftarten und Legenden individuell angepaßt werden.*

Die Leitung eines Projekts ist eine interessante Arbeit, vor allem unter dem Gesichtspunkt, daß sie unterbrechungsgesteuert ist. Bei der Erfahrung und dem Wissen, daß unser Projekt-Team hat, brauchen wir ihnen nicht genau zu sagen, wie sich das Projekt entwickeln soll. Wir brauchen nur zuzuschauen und uns ihre Entscheidungen ansehen. Wir können dann hergehen und sicherstellen, daß nicht einer zu wenig ist oder - was häufiger vorkommt - daß eine andere Gruppe etwas entdeckt hat, was zwischen den Gruppen vermittelt werden muß. Sie müssen verstehen, daß wenn eine Gruppe Probleme mit den Ressourcen hat, wir diese Ressourcen beschaffen können. Wenn wir die Planung ändern müssen, sehen wir warum und es wird dem Projekt-Team mitgeteilt.

MSJ: *Gibt es eine Phase bei diesem Projekt, die sich als besonders schwierig herausstellte?*

Die schwierigste Phase ist, keine falsche Hoffnungen zu wecken. Es besteht zu Beginn eines Projekts die Tendenz, so viel wie möglich zu machen. Wenn man das Produkt auch ausliefern will, muß man das mit der verfügbaren Zeitspanne in Einklang bringen. Die Entwicklung von Microsoft Excel ist nicht beendet. Wir haben noch Hunderte von Features, die wir in das Produkt hineinstecken möchten.

Ein Teil des Jobs bestand darin, sich die lange Liste der möglichen Features anzusehen und zu sagen: »Das ist heute nicht wichtig..., aber das.« Das ist der bei weitem schwierigste Teil des Jobs, sicherzustellen, daß man das richtige tut und das Team auf realistische Ziele ausrichtet.

MSJ: *Hat es irgendwelche Features gegeben, die außer Kontrolle gerieten?*

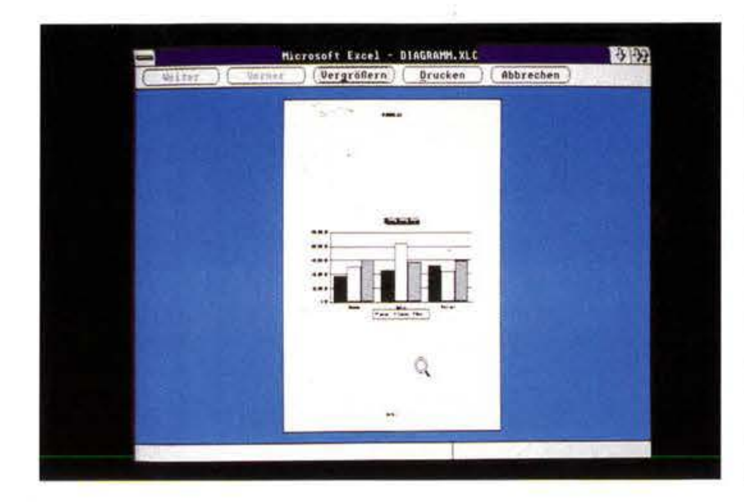
Es gab eigentlich nichts, was außer Kontrolle geriet, denn wir hatten den Vorteil, das erfahrene Leute an diesem Projekt arbeiteten. Sie haben schon gesehen, was es bedeutet zu sagen: »Oh, das ist doch einfach« und dann acht mal länger als erwartet dafür zu benötigen. Es besteht die Tendenz zu einer ausführlicheren Analyse und dem Verständnis, worum es geht, bevor man sich auf etwas einläßt.

MSJ: *War die Nähe von zwei anderen Gruppen, der Windows- und der OS/2-Gruppe, für die Entwicklung von Excel hilfreich?*

Es war aus mehreren Gründen sehr hilfreich, die Windows-Gruppe in der Nähe zu haben. Zunächst einmal ermöglichte es uns, über die Schnittstellen zu diskutieren. Einige der Änderungen, die für Windows 2.0 durchgeführt wurden, sind das Ergebnis von notwendigen Änderungen der Schnittstelle. Das Multiple Document Interface (MDI) ist ein Ergebnis der Arbeit von uns und anderer unabhängiger Softwareentwickler. Wir hielten sie für notwendig, um den Bildschirmbereich für den Benutzer maximal auszunutzen.

Wir konnten das Design von Windows auf ähnliche Weise beeinflussen, wie wir auf die Macintosh-Schnittstelle einwirken konnten. Wir waren aktiv an der Entwicklung komplexer Produkte beteiligt und hatten viele Tips, wie man es besser machen konnte. Die Tatsache, daß sie sich »auf der anderen Seite des Ganges« befanden, war nicht so wichtig - wir hätten es auch über Telefon oder mit Besuchen erledigen können, wie wir das bei Apple gemacht haben.

Die Nähe half uns jedoch, Fehler aufzuspüren. Wir konnten der Windows-Gruppe helfen. Wenn wir auf einen Fehler stießen und glaubten, daß er sich in ihrem Code befindet, konnten wir ihn isolieren und einen Windows-Experten kommen lassen, der es sich ansah und das Problem ausfindig machen konnte.



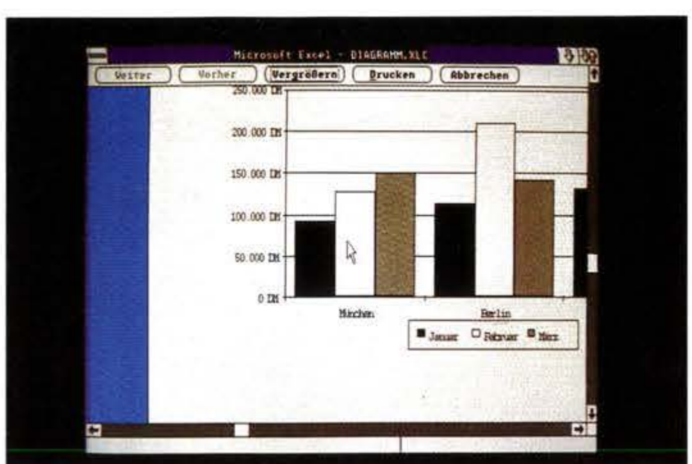


Bild 10: *Man kann sich ein Diagramm oder eine Tabelle vor dem Ausdruck ganzseitig ansehen. Die Lupe ermöglicht es dem Benutzer, sich beliebige Bereiche der Seite vergrößert anzusehen.*

Bild 11: *Mit der Zoom-Möglichkeit kann der Anwender jede Einzelheit eines Diagramms oder einer Tabelle untersuchen. Wenn »Zoom« angeklickt wird, wird zwischen verkleinerter und vergrößerter Darstellung umgeschaltet.*

Das hat sich mehr als alles andere ausgezahlt, daß wir uns den Code ansehen konnten und Hilfe hatten und natürlich die Kommunikation zwischen beiden Gruppen.

MSJ: *Welche Rolle spielte Bill Gates bei der Entwicklung von Microsoft Excel?*

Bill war der Weitsichtige. Er ist einer von den Leuten, die wirklich sagen können: »Wir brauchen ein Feature wie dieses, das ist sehr wichtig.« Er war an dem beteiligt, was aus dem Produkt wurde, weil er sehr viel Überzeugungskraft hat und seine Ideen gut an andere weitergeben kann.

Wir haben jetzt so lange mit Bill zusammengearbeitet, daß wir nun viele derselben Fragen stellen können, die Bill stellen würde. Wir können den richtigen Entwurf machen, weil wir kritischer mit uns sind. Bill hat uns im Laufe der Zeit gezeigt, daß wir nicht selbstzufrieden sein und mehr über das nachdenken sollten, was wir tun. Wenn wir eine Besprechung hatten, stellte er Fragen, die wir dumm fanden, weil wir keine Antworten darauf hatten. Wir haben nun genügend ähnliche Situationen erlebt, so daß wir uns heute dieselben Fargen stellen und sie beantworten können, bevor wir in eine Besprechung gehen.

MSJ: *Gibt es rückblickend etwas, was Sie heute anders machen würden?*

Es wäre schön gewesen, wenn einige Dinge reibungsloser abgelaufen wären, doch es gibt eigentlich nichts, was wir anders machen würden. Dieses Projekt wurde mit einigen neuen Philosophien und neuen Vorgehensweisen für die Entwicklung organisiert. Während die Entwicklung nicht immer so reibungslos verlief, wie wir das gehofft hatten (ganz einfach weil das meistens so ist), verlief sie doch genauso, wie wir hofften, wenn man sich das Ergebnis ansieht und wie das Projekt gemanaget wurde. Wir werden weiter an dieser Vorgehensweise arbeiten.

MSJ: *Wie fühlt sich die Gruppe jetzt, wo eine neue Variante von Microsoft Excel auf dem Markt ist?*

Die ist erst der Beginn von Microsoft Excel. Wir haben gesehen, wie es auf dem Macintosh aussieht und wir haben gesehen, wie es unter Windows aussieht. Noch wichtiger, wir haben so viele Ideen und so viele Neuheiten, die wir implementieren möchten, daß es eigentlich so aussieht, wie der Anfang einer langen Geschichte ausgezeichneter Tabellenkalkulationen.

Die maximale Baudrate des Windows-COM-Treibers

F: *Kann der COM-Treiber von Windows 2.03 auf 19200 Baud eingestellt werden?*

A: Der Windows-Gerätetreiber unterstützt 19200 Baud und überträgt Daten mit dieser Rate. Doch wird die CPU selbst wahrscheinlich die Daten nicht übertragen können, der Durchsatz liegt also mehr im Bereich 9600 Baud.

Prüfsummenberechnung für den Header einer PAINT-Datei

F: *Das Dateiformat von Microsoft Windows Paint enthält im Header ein Feld mit dem Namen* wCheck, *daß eine Prüfsumme enthält. Wie wird sie berechnet?*

A: Die ersten 12 Worte im Header werden mit XOR verknüpft um die Prüfsumme zu erhalten. Eine Methode zur Berechnung der Prüfsumme kann so aussehen:

```
WORD *pfilehdr;
FILEHDR filehdr;

/* Auf das erste Wort initialisieren */
filehdr.wCheck = filehdr.key1;

pfilehdr = &filehdr.key2;
for (i=1; i < 12; i++)
    filehdr.wCheck ^= *pfilehdr++;
```

Ein Zeiger auf den Stack

F: *Wie kann ich einen Zeiger auf den Stack erhalten?*

A: C pusht seine Parameter in umgekehrter Reihenfolge. Der erste Parameter ist immer der zuletzt gepushte, ist immer oben auf dem Stack und hat immer die selbe Adresse relativ zum Beginn der Parameter. Sie können die Stackadresse so festzustellen:

```
far foo()
int x;
{
    int far *y = &x;
}
```




MICROSOFT EXCEL ODER DIE LIEBE ZUR TABELLENKALKULATION.

1. Aufzug, 1. Szene: Anwender, Entscheider, PC und Microsoft Excel.

Anwender (enttäuscht): Grau, teurer Freund, ist alle Theorie ...

PC (hoffnungsvoll): Nicht doch, sieh', hier naht Microsoft Excel schon. Das bringt Aktion in die Tabellenkalkulation.

Microsoft Excel: Kompetent, intelligent und exzellent, steh' ich fürs Zahlenmanagement. Arbeite heute auf Microsoft WINDOWS 2.0 und 386 – morgen, bitte sehr, für den Presentation Manager. Biete dynamischen Datenaustausch und stehe zur Disposition gleichzeitig für viele Tabellen bis hin zur 3. Dimension.

Anwender (beeindruckt): Und wie haltet Ihr's mit Applikation?

Microsoft Excel: Meine Makros machen mich beweglich und dadurch einfach alles möglich. Allerorten – in deutschen und in ganzen Worten.

Anwender (erstaunt): 286/386 Prozessoren? Problembezogene Menüs und Dialogboxen? Makrorekorder, verschiedene Schrifttypen? Übersetzung von Tabellen, Makros und 1-2-3-Befehlen?

Microsoft Excel: Aber ja, was soll die Frage? Mache jeden Auftrag ohne Klage.

Entscheider (überzeugt): Diese weiten Möglichkeiten, was für Zeiten, was für Zeiten. Tabellenkalkulation für jeden Zweck. Formatieren, gestalten, drucken, präsentieren. Funktionen für Grafik und Datenbank. Gestaltungsmöglichkeiten in Farbe. Fürwahr, fürwahr, ich sehe die Entscheidung klar. Schluß jetzt mit den Unklarheiten, und auf in neue große Zeiten.

Microsoft Excel: Die Zukunft ist's, für die ich stehe. Daß Zukunft heute schon geschehe.

Vorhang/Frenetischer Beifall

Tabellenkalkulation mit Graphik- und Datenbankfunktionen – Makroprogrammiersprachen – Kontrollfunktionen – Hilfefunktionen – Computer Based Training – Voraussetzungen: WINDOWS 2.0/386, 286/386 Prozessoren.

MS/DOS CBT   286/386





COUPON

Bitte senden Sie mir Informationsmaterial zu Microsoft Excel. Ich nutze Software: ☐ privat ☐ beruflich/Branche ______

Mein Rechner: ☐ MS-DOS ☐ MS-OS/2 ☐ Macintosh

Bitte senden Sie den Coupon an: Microsoft GmbH · Erdinger Landstraße 2 · 8011 Aschheim-Dornach
Absender nicht vergessen.

Interrupte unter Windows

F: *Welche Interrupte werden von Windows abgefangen, welche werden verwendet?*

A: Windows fängt folgende Interrupte ab:

1. Interrupt 21h: Jeder Funktionsaufruf, der mit Zeichen-Ein-/-ausgaben nach STDIN, STDOUT und STDERR zu tun hat, z.B. IOCTL, OPEN, READ, WRITE usw.
2. BIOS-Interrupt 10h: Jede Funktion, die mit Zeichen-Ein-/-ausgabe zu tun hat. Teilweise TOPVIEW-Emulation.
3. Interrupt 20h, 21h, 24h: Jede Funktion, die mit EXEC, ALLOC und REALLOC zu tun hat.

Windows verwendet nur den Interrupt 3Fh, der Interrupt für dynamisches Linken.

Windows direkt verlassen

F: *Ist es für eine Anwendung unter Windows möglich, eine Meldung an Windows zu schicken, die besagt, daß die Anwendung und Windows beendet werden sollen?*

A: Diese Möglichkeit wird für die aktuellen Versionen von Windows von Microsoft nicht unterstützt. Die folgenden Anweisungen scheinen jedoch zu funktionieren und alle Vektoren wiederherzustellen:

```
DisableOemLayer();
ExitKernel(0);
```

Eine bessere Methode besteht darin, dem MS-DOS Executive mit `PostMessage` die Meldung `WM_SYSCOMMAND` zu schicken. Der Parameter `wParam` dieser Meldung sollte der vordefinierte Wert `SC_CLOSE` sein.

Windows-Utility FreeMem:

Anzeige des freien Speichers unter Windows

Für Programmierer, die sich das erste Mal mit Microsoft Windows auseinandersetzen, scheinen sogar einfache, nichts ausführende Windows-Programme unglaublich lang und komplex zu sein. Sie werden glauben, daß alle Windows-Anwendungen monströse Ansammlungen von Programmzeilen sind. Das ist natürlich nicht wahr.

Nehmen Sie das Programm FreeMem, ein komplettes und nützliches Microsoft Windows-Programm mit weniger als 100 Zeilen C-Quellcode. Das Programm FreeMem zeigt in einem Sinnbild am unteren Rand des Windows-Bildschirms die Größe des verfügbaren Speichers an. Der Wert des freien Speichers wird jede Sekunde aktualisiert und stimmt mit dem in der About-Box im MS-DOS-Fenster angezeigten Wert überein.

Die Kürze von FreeMem hilft die Struktur von Windows-Anwendungen verständlicher werden zu lassen, und erlaubt die ausführliche Diskussion einiger Programmierdetails unter Windows. Da FreeMem ein ungewöhnliches Windows-Programm ist, untersuchen wir auch einige seiner Tricks.



Bild 1: *Die Windows-Utility FreeMem.*

Gesamtstruktur

FREEMEM.C enthält nur zwei Funktionen: WinMain (*Listing 1*) und WndProc (*Listing 2*). Ähnliche Funktionen finden Sie in den meisten Windows-Anwendungen.

WinMain ist der Eintritt zu FreeMem. WinMain führt hauptsächlich alle Initialisierungen durch, die zum Erzeugen und zum Anzeigen des Fensters notwendig sind.

Die an die Funktion RegisterClass übergebene Struktur WndClass definiert die Fensterklasse. Der wichtigste Teil der Struktur WndClass ist lpfnWndProc, welcher die Funktion innerhalb von FreeMem angibt, die die Nachrichten von Microsoft Windows bearbeitet. Dies ist die Funktion WndProc, die Sie im Listing 2 sehen.

Die Funktionen CreateWindow, ShowWindow und UpdateWindow veranlassen alle Fenster zum Senden von Nachrichten an die Funktion WndProc. Innerhalb der Funktion WndProc werden diese Nachrichten durch Namen, die mit WM beginnen, identifiziert. Diese Namen sind einfache Makrobezeichner, die in der Header-Datei WINDOWS.H enthalten sind, und zum leichteren Verständnis die Angabe von Zahlencodes überflüssig machten.

WndProc unterscheidet die Nachrichten durch eine case-Anweisung. Nur einige der von Windows gesendeten Nachrichten werden von WndProc direkt bearbeitet. Die restlichen werden an die in Windows enthaltene Funktion DefWindowProc weitergeleitet, wo eine Standardbearbeitung stattfindet.

CreateWindow veranlaßt Microsoft Windows, die Nachricht WM_CREATE zu erzeugen. ShowWindow erzeugt in der Regel eine ganze Anzahl von Nachrichten, darunter auch WM_SIZE und WM_ERASEBKGND. Diese Nachrichten sind für das Darstellen des Fensters und das Löschen des Hintergrunds zuständig. Beim Aufruf der FreeMem-Funktion UpdateWindow erzeugt Microsoft Windows eine WM_PAINT-Nachricht, die WndProc auffordert, den Client-Bereich des Fensters darzustellen.

Nach dem Aufruf von UpdateWindow verweilt FreeMem in einer Nachrichtenschleife. Der Funktionsaufruf GetMessage erhält eine Nachricht von FreeMems Nachrichtenschlange. Sind keine Nachrichten für FreeMem vorhanden, kann die Kontrolle an eine andere Windows-Anwendung übergeben werden. Die derzeitige nicht zeitscheibengesteuerte Ausführung von Windows garantiert, daß nur ein Aufruf von GetMessage die Ausführung von FreeMem unterbrechen kann. FreeMem erhält die Kontrolle wieder, wenn einige Nachrichten in seiner Schlange sind und die Schlangen der anderen Anwendungen leer sind. In Wirklichkeit ist dieses Thema ein wenig komplexer, aber ich werde es später ausführlicher erörtern.

TranslateMessage übersetzt Tastendrücke in Zeichencode-Nachrichten. Obwohl sich FreeMem nicht um Tastendrücke kümmert, ist diese Schnittstelle notwendig für das System-Menü von FreeMem. DispatchMessage sendet die Nachrichten dann an die Funktion WndProc.

FreeMem ist beendet, wenn GetMessage eine Nachricht WM_QUIT aus der Nachrichtenschlange erhält. Die Funktion GetMessage liefert in diesem Fall den Wert 0 zurück, und FreeMem entfernt sich aus der Nachrichtenschlange.

Soweit entspricht diese Funktion den meisten Windows-Anwendungen, aber sehen wir uns die Details näher an.

Nützlicher Gebrauch der Sinnbilder

Anders als bei den meisten Windows-Programmen ist bei FreeMem nur die Darstellung als Sinnbild beabsichtigt. Nach dem Start von FreeMem können Sie die Tastatur oder die Maus zum Öffnen des Sinnbilds in ein reguläres geteiltes Fenster benutzen, aber Sie erhalten dadurch nicht mehr Informationen.

Eine große Zahl Windows-Programme besitzt statische bildhafte Sinnbilder. Die Programmierer erzeugen diese Sinnbilder in der Regel mit dem Diensprogramm ICONEDIT, das im Microsoft Windows Software Development Kit enthalten ist. Die Sinnbilddatei (.ICO) wird in der Ressourcendatei durch einen Namen angegeben. Beim Erzeugen der Fensterklasse wird das Sinnbild durch das Statement

```
WndClass.hIcon = LoadIcon(hInstance, (LPSTR)szAppName);
```

angegeben. Die Funktion LoadIcon lädt das Sinnbild aus dem Teil der EXE-Datei, wo alle Ressourcen angesiedelt sind, und weist ihr eine Handle zu. (Ich setze hier voraus, daß das Sinnbild denselben Namen wie die Anwendung trägt, und daß szAppName ein Zeiger auf eine Zeichenkette mit diesem Namen ist.)

Wenn Sie statt dessen das Statement

```
WndClass.hIcon = NULL;
```

benutzen, dann ist die Anwendung selbst für das Darstellen des Sinnbilds zuständig. Das gibt Ihrer Anwendung die Möglichkeit, ein dynamisches Sinnbild zu erzeugen, das Sie ändern können. Die UHR-Anwendung in Microsoft Windows benützt dieselbe Technik, um die Zeit anzuzeigen, sogar dann, wenn das Fenster als Sinnbild dargestellt wird.

Aus der Sicht des Programmierers ist ein NULL-Sinnbild ein kleines Fenster, in das in derselben Art geschrieben werden kann, wie in den Client-Bereich eines normalen Fensters. Wenn Sie wissen müssen, wann Ihre Anwendung als Sinnbild dargestellt ist, können Sie diese Information der Nachricht WM_SIZE entnehmen. UHR wertet diese Änderung aus, kann so den Sekundenzeiger unterdrücken und aktualisiert die Darstellung in der Sinnbildform dann nur noch jede Minute.

Sinnbilddarstellung erzwingen

Um eine Windows-Anwendung im MS-DOS-Fenster zu starten, drücken Sie entweder die [Return]-Taste während der Cursor auf dem Programmnamen steht, klicken den Programmnamen zweimal mit der Maus an oder wählen File Run aus dem Menü. Wollen Sie statt dessen ein Programm als Sinnbild laden, so drücken Sie [Shift][Return], wenn der Cursor auf dem Programmnamen steht oder wählen File Load aus dem Menü.

```
/* FreeMem.C -- Windows-Anwendung,
   die den freien Speicher anzeigt */

#include    <windows.h> /* Alle Windows-Funktionen */
#include    <stdlib.h>  /* itoa */
#include    <string.h>  /* strcat und strlen */

long FAR PASCAL WndProc(HWND, unsigned, WORD, LONG);


int PASCAL WinMain(hInstance, hPrevInstance, lpszCmdLine,
                   nCmdShow)
    HANDLE hInstance, hPrevInstance;
    LPSTR  lpszCmdLine;
    int    nCmdShow;
    {
    static char szAppName[] = "FreeMem";
    WNDCLASS    WndClass;
    HWND        hWnd;
    MSG         msg;

    if(hPrevInstance)
        return(FALSE);


    WndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
    WndClass.hIcon = NULL;
    WndClass.cbClsExtra = 0;
    WndClass.cbWndExtra = 0;
    WndClass.lpszMenuName = NULL;
    WndClass.lpszClassName = (LPSTR) szAppName;
    WndClass.hbrBackground =
                     (HBRUSH) GetStockObject (WHITE_BRUSH);
    WndClass.hInstance = hInstance;
    WndClass.style = CS_HREDRAW | CS_VREDRAW;
    WndClass.lpfnWndProc = WndProc;

    if(!RegisterClass((LPWNDCLASS) &WndClass))
        return(FALSE);


    hWnd = CreateWindow((LPSTR) szAppName,
                     (LPSTR) szAppName,
                     WS_TILEDWINDOW, 0, 0, 0, 0,
                     (HWND) NULL, (HMENU) NULL,
                     (HANDLE) hInstance,
                     (LPSTR) NULL);

    if(!SetTimer (hWnd, 1, 1000, NULL))
        return(FALSE);

    ShowWindow(hWnd, SHOW_ICONWINDOW);
    UpdateWindow(hWnd);

    while(GetMessage((LPMSG) &msg, NULL, 0, 0))
    {
        TranslateMessage((LPMSG) &msg);
        DispatchMessage((LPMSG) &msg);
    }
    return((int) msg.wParam);
    }
```

Listing 1: *Die erste Hälfte von FreeMem enthält die Funktion* WinMain, *in der die notwendige Initialisierung erfolgt und die die Meldungsschleife enthält.*

Bei FreeMem können Sie allerdings tun, was Sie wollen - es wird immer als Sinnbild dargestellt. Die meisten Windows-Programme rufen kurz vor dem Eintritt in die Nachrichtenschleife folgende Funktion auf:

```
ShowWindow(hWnd, nCmdShow);
```

```
long FAR PASCAL WndProc(hWnd, message, wParam, lParam)
    HWND         hWnd;
    unsigned     message;
    WORD         wParam;
    LONG         lParam;
    {
    static int   mem, lastmem;
    char         buffer[20];
    PAINTSTRUCT  ps;
    RECT         rect;

    switch(message)
        {
        case WM_TIMER:
            mem = (int) (GlobalCompact (0L) / 1024);
            if(mem != lastmem)
                InvalidateRect(hWnd, NULL, TRUE);
            lastmem = mem;
            break;

        case WM_PAINT:
            BeginPaint(hWnd, (LPPAINTSTRUCT) &ps);
            GetClientRect(hWnd, (LPRECT) &rect);
            DrawText(ps.hdc, (LPSTR) buffer,
                     strlen(strcat(itoa(mem,
                                buffer, 10), "K Frei")),
                     (LPRECT) &rect, DT_WORDBREAK);
            EndPaint(hWnd, (LPPAINTSTRUCT) &ps);
            break;

        case WM_DESTROY:
            KillTimer(hWnd, 1);
            PostQuitMessage(0);
            break;

        default:
            return(DefWindowProc(hWnd,message, wParam,lParam));
        }
    return((long) 0L);
    }
```

Listing 2: *Die zweite Hälfte von FreeMem enthält die Fensterprozedur mit dem Namen* `WndProc`. `WndProc` *verarbeitet die Meldungen, die Windows an das Fenster schickt.*

Die Variable `nCmdShow` wird dem Programm als Parameter an `WinMain` übergeben. Wenn Sie ein Programm vom MS-DOS-Fenster aus laufen lassen, wird `nCmdShow` durch die Handle des Fensters ersetzt, das die Anwendung auf dem Bildschirm darstellt, wobei es sich normalerweise um das MS-DOS-Fenster handelt. Laden Sie aber eine Anwendung als Sinnbild, so wird `nCmdShow` gleich `SHOW_ICONWINDOW` gesetzt. Ihr Programm muß dies normalerweise nicht unterscheiden, sondern es gibt diesen Parameter einfach an `ShowWindow` weiter.

Sie brauchen normalerweise die Variable `nCmdShow` nicht in Verbindung mit `ShowWindow` zu benutzen. In FreeMem verwende ich statt dessen die Zeile

```
ShowWindow(hWnd, SHOW_ICONWINDOW);
```

Das erzwingt die Sinnbild-Darstellung des Fensters, unabhängig von der Variablen `nCmdShow`. Sie können mit dieser Technik auch andere Ergebnisse erzielen. Wenn eine Anwendung immer in Ganzbildschirm-Darstellung starten soll, können Sie folgendes Statement benutzen:

```
ShowWindow(hWnd, SHOW_FULLSCREEN);
```

Sie können FreeMem sogar veranlassen eine bestimmte Sinnbildposition am unteren Rand des Bildschirms zu benutzen, indem Sie den Aufruf `ShowWindow` in FreeMem durch folgenden ersetzen:

```
ShowWindow(hWnd, (int) 0xFF8F);
```

Das Sinnbild wird an Position 15 positioniert, ganz am rechten Ende des Bildschirms. Die Syntax ist etwas sonderbar, aber im Programmer's Reference Manual des Software Development Kits dokumentiert.

Sinnbilddarstellung fortsetzen

Wie ich oben schon angedeutet habe, können Sie ohne weiteres die Tastatur oder die Maus zum Öffnen des FreeMem-Sinnbilds in ein reguläres geteiltes Fenster benutzen. Wenn Sie dies aber verhindern und sicherstellen wollen, daß FreeMem immer nur als Sinnbild dargestellt wird, so brauchen Sie nur die beiden Zeilen in die Funktion `WndProc` einfügen:

```
case WM_QUERYOPEN:
     break;
```

Ein sinnvoller Platz wäre unmittelbar vor der Stelle

```
case WM_DESTROY:
```

Wenn Sie jetzt die Maus benutzen, um das Sinnbild zu öffnen, so springt das Sinnbild einfach wieder an seinen Platz am unteren Bildschirmrand zurück.

Die beiden Zeilen scheinen nicht viel zu tun, aber ein genauerer Blick löst das Rätsel. Microsoft Windows schickt die Nachricht `WM_QUERYOPEN` an ein Programm, wenn es ein Sinnbild öffnen will. Die Dokumentation besagt zu `WM_QUERYOPEN`, daß eine Windows-Funktion, wie etwa `WndProc`, 0 zurückgeben muß, um das Sinnbild vom Öffnen abzuhalten. Unter normalen Umständen wird die Nachricht `WM_QUERYOPEN` an die Funktion `DefWindowProc` weitergegeben, die einen Wert ungleich 0 liefert. (Die Funktion `DefWindowProc` ist in der Datei `WINDWP.C` im Windows Software Development Kit enthalten.) Windows öffnet dann das Sinnbild.

Mit den beiden oben gezeigten Zeilen aber gibt `WndProc` den Wert 0 für die Nachricht `WM_QUERYOPEN` zurück. Somit antwortet `WndProc` auf die Frage von Windows, ob es geöffnet werden will, mit »Null«, was in diesem Falle »Nein Danke« bedeutet.

NEU: MS WINDOWS 2.0

Ein erfolgreiches Buch...

- te-wi's Bestseller-Titel WINDOWS völlig überarbeitet für Version 2.0
- Zeigt die gesamte WINDOWS-Information in 69 leicht lesbaren Modulen
- Zum Erlernen wie ein WINDOWS-2.0-Kurs lesbar; danach als WINDOWS-2.0-Lexikon

WINDOWS 2.0: Einführung + Referenz
Whitsitt/Bryan
496 Seiten. Hardcover. DM 79,–

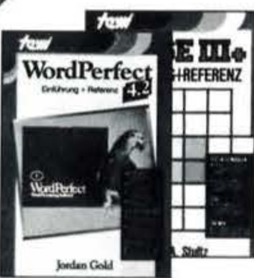


...und seine Leser

- für WINDOWS-2.0-Beginner ein idealer und erprobter Selbstlerntext
- für WINDOWS-2.0-Kenner ein ‚Lexikon neben dem Computer'
- für WINDOWS-1.0-Benutzer bleibt der Vorgängertitel ‚WINDOWS' zu empfehlen
- Auch als Einführung für WINDOWS/386 geeignet.

tewi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Weitere te-wi-Bücher



WordPerfekt 4.2: Einführung + Referenz
(J. Gold)
Gliederung in 64 Module, zur Einführung als Kurstext lesbar, dann als alphabetisches Befehlslexikon. Viele praktische Beispiele.
496 Seiten. Hardcover. DM 69,–

dBase III+: Einführung + Referenz
(Stultz)
Update des dBASE-III-Bestsellers. Kurstext und Lexikon in 70 Modulen.
480 Seiten. Hardcover. DM 79,–



Star Writer PC 3.0 Einführung + Handbuch
(Franz Grieser)
Dieses Buch formuliert alltägliche und seltene Fragen an das System und beantwortet sie in sofort am PC anwendbarer Form.
Ca. 300 Seiten. Hardcover. DM 49,–

Geld + Taschencomputer
(H. Elster)
Wie man Geldfragen auf Taschencomputern löst: Auto, Kredit, Immobilien, Preise, Rechnungen usw.
Ca. 250 Seiten. Hardcover. DM 49,–



TURBO PASCAL Systematisch: Teil 1: Einführung in Sprache und Anwendung
(Ciric/Thies)
Band 1 eines Pascal-Kurses für Auszubildende der Industrie. Systematisch, PC-orientiert, mit Musterprogrammen, Übungen und Compilerpraxis.
496 Seiten. Softcover. DM 49,–

DAS „C"-BUCH
(Herold/Unger)
Ein „C"-Kurs der Industrie. Für sämtliche C-Konstrukte. Über 100 Beispiele. Anspruchsvoll in Text/Bildmaterial.
576 Seiten. Softcover. DM 79,–



DESKTOP KNIGGE Setzerwissen für Desktop-Publisher
(Philipp Luidl)
Ein bekannter deutscher Typograph sammelte für alle ernsthaften Desktop-Publisher das traditionsreiche Berufswissen von Setzern, Druckern und Graphikern.
Ca. 250 Seiten. Hardcover. DM 79,–

VENTURA PUBLISHER 1.1: Einführung + Referenz
(R. M. Hohol)
Überlegt gegliederte, vollständige VENTURA 1.0/1.1-Darstellung.
464 Seiten. Hardcover. DM 79,–



AutoCAD: Versionen 2.5, 2.6 und 9.0 Einführung + Referenz
(Berghauser/Schlieve)
Kurstext und Lexikon in 93 Modulen. Pragmatische Einführung in AutoCAD's Leistungen und Musteranwendungen.
480 Seiten. Hardcover. DM 89,–

TURBO AutoCAD
(Schaefer/Brittain)
Themen: Makros für eigene AutoCAD-Befehle; automatisierte Zeichnungsabläufe; selbstgestaltete Tablett/Bildschirmmenüs.
422 Seiten. Hardcover. DM 79,–



Festplattenverwaltung:
(Dietzel)
Professionelle Verwaltung von Daten und Programmen auf Platten mit einem Verwaltungsprogramm.
192 Seiten. Softcover. DM 39,–

MS DOS Einfache Zugänge
(Fürst)
Spotlights auf MS DOS für eilige PC-Benutzer. Befehle in sofort benutzbarer Form.
162 Seiten. Softcover. DM 39,–

tm 5197/1

Noch im Programm: PC-Software:
MS DOS, Wordstar, Multiplan, dBASE, DM 59,–
Das 8086/8088-Buch, DM 79,–

Für Eilbestellungen: Hotline 0 21 91-34 20 77

```
if(!SetTimer (hWnd, 1, 1000, NULL))
        {
        MessageBox(hWnd, "Zu viele Timer!", NULL, MB_OK);
        return FALSE;
        }
```

Listing 3: *Dieser alternative Programmteil teilt dem Benutzer mit, warum das Programm beendet werden muß, wenn* `SetTimer` *0 zurückgibt.*

Die Timer-Nachrichten

FreeMem aktualisiert seine Anzeige auch wenn andere Programme laufen. Es führt dies mit der Hilfe des Timers von Microsoft Windows durch, der die Nachricht `WM_TIMER` an die Fensterfunktion sendet. In dieser Beziehung gleicht FreeMem der UHR-Anwendung. Der Timer erlaubt eine Art von Multitasking, ohne daß die Anwendung wertvolle Prozessorzeit verschwendet.

In der Funktion `WinMain` fordert FreeMem mit folgendem Statement von Windows einen Timer an:

```
if(!SetTimer(hWnd, 1, 1000, NULL))
    return(FALSE);
```

Der dritte Parameter `1000` im Funktionsaufruf von SetTimer bedeutet, daß FreeMem alle 1000 Millisekunden bzw. jede Sekunde eine `WM_TIMER`-Nachricht erhalten möchte.

Wenn `SetTimer` 0 zurückgibt, bedeutet dies, daß kein Timer mehr für dieses Programm zur Verfügung steht. Wenn Sie je probiert haben, wie viele UHR-Programme gleichzeitig unter Windows laufen können, wissen Sie, daß das Maximum 15 ist. Jeder weitere Aufruf von `SetTimer` gibt 0 zurück. Tatsächlich gibt es einen Weg, von Windows mehr als 15 Timer zu erhalten, der ist aber etwas komplizierter.

Wenn FreeMem keinen Timer erhält, muß die Funktion `WinMain` den Wert 0 (Wert für `FALSE`) zurückgeben, was zum Programmabbruch führt. Wie Sie an den beiden anderen Zeilen mit `return (FALSE)` in `WinMain` sehen, wird FreeMem auch beendet, wenn es schon einmal läuft oder wenn es die Fensterklasse nicht registrieren kann. Obwohl es nicht unbedingt notwendig wäre, mehrere Vorkommen von FreeMem nicht zuzulassen, macht es doch nicht viel Sinn.

Wenn Sie möchten, daß FreeMem Ihnen mitteilt, warum es abbricht, wenn kein Timer mehr vorhanden ist, so können Sie den Programmteil aus *Listing 3* verwenden. Dieser Teil ist informativer, aber nicht besonders höflich.

Der zweite Parameter beim Aufruf `SetTimer` wird als Timer-ID bezeichnet. Beim Erhalt der Nachricht `WM_TIMER` bekommt die Funktion `WndProc` im Parameter `wParam` diesen Wert. Durch die Verwendung mehrerer Timer-IDs kann eine Windows-Anwendung mehrere Timer für unterschiedliche Aufgaben einsetzen. FreeMem muß die Timer-ID auch zum Zurückgeben des Timers mittels `KillTimer` verwenden, wenn `WndProc` die Nachricht `WM_DESTROY` erhält.

Die Nachrichten `WM_TIMER` sind nicht asynchron. Die Angabe von 1000 Millisekunden im Aufruf `SetTimer`, garantiert nicht, daß die Fensterfunktion genau jede Sekunde die Nachricht `WM_TIMER` erhält. Die `WM_TIMER`-Nachrichten werden in der normalen Nachrichtenschlange verwaltet und mit den anderen Nachrichten synchronisiert.

Ist eine andere Anwendung länger als eine Sekunde aktiv, so erhält FreeMem während dieser Zeit keine einzige `WM_TIMER`-Nachricht. FreeMem erhält seine nächste `WM_TIMER`-Nachricht erst aus der Schlange, wenn die andere Anwendung die Kontrolle durch einen Aufruf von `GetMessage`, `PeekMessage` oder `WaitMessage` abgibt.

Tatsächlich werden `WM_TIMER`-Nachrichten ebenso wie `WM_PAINT`-Nachrichten von Microsoft Windows mit niedriger Priorität gehandhabt. Das bedeutet, daß die Kontrolle an andere Programme nur abgegeben wird, wenn die Nachrichtenschlange des Programms leer ist. In Wirklichkeit aber wird die Kontrolle an ein anderes Programm übergeben, wenn das aktuelle nur `WM_PAINT`- oder `WM_TIMER`-Nachrichten in seiner Schlange enthält, das andere aber irgendeine andere Nachricht, außer diesen beiden.

Des weiteren fügt Windows keine zusätzlichen `WM_TIMER`-Nachrichten in die Schlange ein, wenn eine andere Anwendung läuft. In diesem Fall faßt Windows mehrere `WM_TIMER`-Nachrichten zu einer zusammen, so daß die Anwendung nicht mehrere davon auf einmal erhält.

Obwohl die Handhabung der `WM_TIMER`-Nachrichten für ein Programm wie FreeMem genügt, sollten Sie diesen Aspekt im Auge behalten, falls Sie je ein Programm wie die Windows-UHR entwickeln. Ein Timer-Wert von 1000 Millisekunden garantiert keine 3600 `WM_TIMER`-Nachrichten je Stunde. Wenn Sie eine `WM_TIMER`-Nachricht erhalten, sollten Sie die wirkliche Zeit entweder durch eine C-Funktion, oder durch MS-DOS selbst in Erfahrung bringen. Sie können die Zeit nicht allein aufgrund von `WM_TIMER`-Nachrichten mitführen.

Berechnung des Speichers

Beim Erhalt der Nachricht `WM_TIMER` muß FreeMem die Größe des freien Speichers bestimmen. Diese kleine Aufgabe stellte die größte Anforderung beim Programmieren von FreeMem. Ich wußte, daß die Information verfügbar ist, da das MS-DOS-Fenster in der About-Box den Wert des freien Speichers anzeigt. Da aber Microsoft Windows ungefähr 400 Funktionen zur Verfügung stellt, ist es ab und zu etwas schwierig, die benötigte Funktion zu finden.

Es stellte sich heraus, daß das MS-DOS-Fenster die Information des freien Speichers durch den Aufruf von

GlobalCompact mit einem Parameter von 0 erhält. Windows-Programme rufen die Funktion GlobalCompact normalerweise auf, um freien Speicher vom globalen Heap zu erhalten. Wenn es notwendig ist, komprimiert Windows die Speicherblöcke und gibt die als discardable markierten Segmente beim Aufruf von GlobalCompact frei. (Der globale Heap setzt sich aus Speicher außerhalb des lokalen Datensegments zusammen.)

Genaueres Nachlesen der Dokumentation zu GlobalCompact ergab, daß im Falle des Parameters 0 die Funktion einen Wert zurückgibt, den Speicher aber nicht verschiebt. So ist der von FreeMem angezeigte Wert wirklich der mögliche freie Speicher, und nicht der aktuelle freie Speicher. Der Wert gibt an, wieviel Speicher eine Windows-Anwendung vom globalen Heap im Bedarfsfalle erhalten könnte.

Wenn Sie mit dem Hilfsprogramm HEAPWALK, das im Software Development Kit enthalten ist, vertraut sind, können Sie eine Menge Zeit damit verbringen, die Werte von FreeMem und HEAPWALK in Übereinstimmung zu bringen. Es scheint einfach nicht einsichtig zu sein. GlobalCompact gibt ungefähr die Größe des Bereichs des freien Speichers in der Mitte des von Windows verwalteten Speicherpools zurück und zusätzlich einige discardable-Segmente über diesem Bereich des freien Speichers. Kurz vor dem discardable Codesegment von FreeMem bricht es aber ab. GlobalCompact kann den Code von FreeMem selbstverständlich nicht auslagern, wenn FreeMem selbst diese Funktion aufruft.

Zeichnen des Sinnbilds

In FreeMem wickelt der Programmteil, der die WM_TIMER-Nachrichten bearbeitet, nicht selbst die Aktualisierung des angezeigten Sinnbilds ab. Statt dessen zeigt folgender Funktionsaufruf

```
InvalidateRect(hWnd, NULL, TRUE);
```

Windows an, daß der Inhalt des Fensters von FreeMem jetzt ungültig ist und deshalb aktualisiert werden muß. Die Funktion InvalidateRect veranlaßt Windows eine WM_PAINT-Nachricht in die Nachrichtenschlange einzufügen. WndProc aktualisiert das Fenster nur beim Erhalt dieser WM_PAINT-Nachricht.

Es gibt Alternativen zu dieser Methode. Statt des Aufrufs InvalidateRect kann WM_PAINT das Fenster direkt aktualisieren. Es ist notwendig, den Anzeige-Kontext zu erhalten mit dem Aufruf:

```
hDC = GetDC(hWnd);
```

```
NAME        FreeMem

DESCRIPTION 'Free Memory Display by Charles Petzold'

STUB        'WINSTUB.EXE'

CODE        MOVEABLE
DATA        MOVEABLE MULTIPLE

HEAPSIZE    1024
STACKSIZE   4096

EXPORTS
        WndProc @1
```

Listing 4: *Die Moduldefinitionsdatei* FREEMEM.DEF *wird für Informationen benötigt, die nicht im Quellcode definiert sind.*

Anschließend sind Aufrufe von GetClientRect und DrawText erforderlich, ähnlich wie in der WM_PAINT-Logik, aber unter der Verwendung von hDC als Anzeige-Kontext, statt ps.hdc. WM_PAINT müßte anschließend den Anzeige-Kontext wieder freigeben:

```
ReleaseDC(hWnd, hDC);
```

Obwohl das sicher korrekt ist, wählte ich einen anderen Weg. WndProc muß auch die WM_PAINT-Nachrichten bearbeiten, die aus anderen Gründen, als dem Aktualisieren des Wertes für den freien Speicher, gesendet werden. Wenn Sie zum Beispiel die Maus zum Verschieben des FreeMem-Sinnbilds verwenden, sendet Windows unter Umständen eine WM_PAINT-Nachricht, damit FreeMem sein Sinnbild erneut darstellen kann. Deshalb muß entweder der Anzeigecode dupliziert oder in eine Funktion verlagert werden.

Der Aufruf von InvalidateRect erledigt dies tatsächlich für uns, indem er die Nachricht WM_PAINT erzeugt. Dies erlaubt es, denselben Anzeigecode für alle Anzeigearbeiten zu verwenden.

Die WM_PAINT-Nachrichten besitzen in Windows eine niedrige Priorität. Sie werden immer an das Ende der Nachrichtenschlange angehängt, und werden aus der Schlange nur geholt, wenn keine andere Nachricht mehr vorhanden ist. Es gibt aber auch einen Ausweg. Nach dem Aufruf von InvalidateRect kann WM_TIMER dann

```
UpdateWindow(hWnd);
```

aufrufen, genau wie wir es in WinMain durchführten. Dies veranlaßt Windows, die WndProc-Funktion direkt mit der Nachricht WM_PAINT aufzurufen, ohne Umweg über die Nachrichtenschlange. Dies ist aber hier wirklich nicht notwendig. Ich möchte FreeMem als Task mit niedriger Priorität arbeiten lassen. Wenn in einer anderen Anwendung etwas los ist, möchte ich nicht, daß FreeMem durch ständiges Neuzeichnen des Sinnbilds Zeit verschwendet.

```
/Make-Datei für FREEMEM
/-----------------------

freemem.obj:    freemem.c
    cl -c -d -Gsw -Os -W2 -Zdp freemem.c

freemem.exe:    freemem.obj freemem.def
    link4 freemem, /align:16, /map/line, slibw, freemem
    mapsym freemem
```

Listing 5: *Diese Datei mit dem Namen* FREEMEM *(ohne Erweiterung) ist die Make-Datei für die Generierung von* FREEMEM.EXE.

Die von FreeMem beim Bearbeiten der Nachricht WM_PAINT benützte Funktion DrawText ist sehr angenehm für einfachen Wortumbruchstext. Bedenken Sie, daß die Zeile mit dem Aufruf DrawText

```
strlen(strcat(itoa(mem, buffer, 10), "K Frei"))
```

denselben Zweck erfüllt, wie der gewohntere Ausdruck

```
sprintf(buffer, "%dK Frei", mem)
```

Sprintf ist jedoch eine große Funktion. Durch die Verwendung von strlen, strcat und itoa können wir die Größe von FREEMEM.EXE um ungefähr 3 Kbyte reduzieren. Nicht viel, aber doch etwas.

Die Erzeugung von FreeMem

Neben dem Quellcode von FREEMEM.C benötigen Sie noch zwei andere Dateien für die Erzeugung von FreeMem. Die erste Datei ist die Moduldefinitionsdatei FREEMEM.DEF (*Listing 4*). Die Datei FREEMEM.DEF enthält die Standard-Informationen, wie sie für die meisten kleinen Windows-Anwendungen typisch sind.

Das Programm WINSTUB.EXE, das in der Zeile STUB in FREEMEM.DEF angegeben ist, ist im Software Development Kit enthalten. Dieses Programm läuft, wenn Sie FreeMem außerhalb von Windows aufrufen. Es zeigt einfach die Meldung »This Programm requires Microsoft Windows« an.

Wenn Sie wollen, können Sie ein normales MS-DOS-Programm erzeugen, das den freien Speicher ähnlich dem von CHKDSK errechneten Wert anzeigt. Wenn Sie sich entscheiden, dieses normale MS-DOS-Programm zum Beispiel DOSFREE.EXE zu benennen, können Sie in der Datei FREEMEM.DEF angeben.

```
STUB    'DOSFREE.EXE'
```

Dieser kleine Trick gestattet die Verwendung von FreeMem sowohl auf der MS-DOS-Kommandoebene, als auch unter Windows. FREEMEM.EXE enthält dann zwei verwandte, aber ziemlich unähnliche Programme in einer Datei. Die MAKE-Datei von FreeMem (die einfach FREEMEM ohne Erweiterung heißt) ist im *Listing 5* zu sehen. Wenn Sie den Befehl

```
MAKE FREEMEM
```

ausführen, wird der Quellcode von FREEMEM.C übersetzt und mit den zugehörigen Windows und C-Bibliotheken und den Informationen aus der Definitionsdatei FREEMEM.DEF gelinkt.

Keine Ressourcendatei?

Im Gegensatz zu vielen einfachen Windows-Anwendungen benötigt FreeMem keine Ressourcendatei (eine Datei mit dem Zusatz .RC). FreeMem verfügt über kein Menü, kein bildhaftes Sinnbild und keine Dialogboxen. Die einzige Anwendung, die wir für eine Ressourcendatei hätten, wäre die zum Speichern von Zeichenketten. Die einzige Zeichenkette, die FreeMem benötigt, ist aber der Text mit dem Wort Free.

Für längere Programme ist die Verwendung einer Ressourcendatei dringend angeraten, um alle Textstrings zu speichern, da hierdurch eine Übersetzung des Programms in eine andere Sprache vereinfacht wird. Für ein kleines Programm stellt das Laden von Zeichenketten aus der Ressourcendatei aber eine Last dar. FreeMem ist klein genug, daß das Übersetzen kein großes Problem darstellt, und man wäre päpstlicher als der Papst, würde man für diesen einzigen Textstring eine Ressourcendatei verwenden.

Nutzen in der Entwicklung

Ich mag FreeMem gern unter Windows geladen und laufen haben, nur um mich aufmerksam zu machen, wann Windows der Speicher zu eng wird. Aber der Nutzen bei der Programmentwicklung ist selbstverständlich größer.

Wenn Sie eine Windows-Anwendung entwickeln, richten Sie während des Testens einen Blick auf das FreeMem-Sinnbild. Sollte die Größe des freien Speichers kleiner werden, haben Sie vermutlich vergessen, allokierten Speicher im Programm wieder freizugeben. Sie müßten dann die Utility HEAPWALKER zu Rate ziehen, um diese verlorenen Speicherblöcke ausfindig zu machen.

Bedenken Sie aber, daß FreeMem vor und nach dem Programmlauf einer Anwendung nicht den gleichen Wert anzeigt. Die Komplexität der Speicherverwaltung von Windows ist für diesen Unterschied bei den meisten großen Programmen verantwortlich.

Charles Petzold

Ein Dateiformat für den Grafikaustausch:

TIFF: Der neue Grafikstandard

In den Anfangstagen der Microcomputer waren die Anwender schon froh, wenn sie ein Zeichenprogramm besaßen, das elementare Zeichenoperationen unterstützte. Sie konnten einfache Zeichnungen für Dokumente anfertigen, diese ausdrucken und sie per Hand einfügen. Leider sah das Ergebnis amateurhaft aus und war oftmals fast nicht akzeptabel.

Die Schwierigkeit bestand darin, daß keine mächtigeren Werkzeuge vorhanden waren, die die Zeichenwünsche befriedigten. Was war, wenn die Anwender vorhandene Fotografien benutzen wollten? Sie brauchten eine Möglichkeit, die professionellen Zeichnungen zu digitalisieren und sie dann elektronisch in die Dokumente einzufügen.

Das änderte sich mit der Entwicklung der Scanner. Die meisten verfügbaren Scanner können vorhandene Kunstwerke oder Fotografien mit einer Auflösung von 300 Punkten/Zoll (dpi) abtasten. Das klingt nach der Erfüllung von Träumen, aber es fehlt noch ein Mosaiksteinchen. Die Anwender benötigen noch ein Standard-Dateiformat, um jeden beliebigen Scanner zum Digitalisieren von Abbildungen verwenden zu können, diese dann mit jedem Zeichen- oder Grafik-Programm ihrer Wahl bearbeiten zu können, und dann das Abbild elektronisch in das Dokument einfügen zu können. Wenn zu guter Letzt der ganze Vorgang noch in kurzer Zeit zu bewältigen wäre, würde die Erfüllung des Traums wahr.

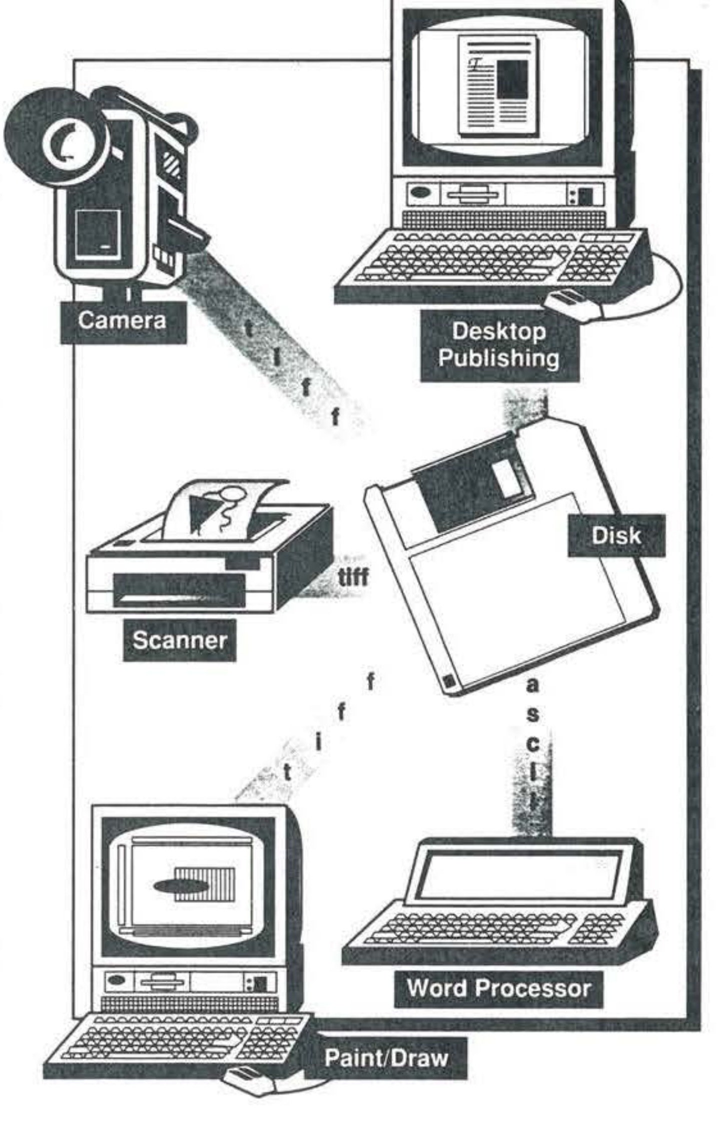


Bild 1: *TIFF kann aufgrund seiner Fähigkeiten mit Scannern, Zeichenprogrammen und Kameras verwendet werden. Grafiken einfacher Zeichenprogramme werden genauso gehandhabt, wie komplexe medizinische Dokumente.*

Was ist TIFF?

Die Aldus Corporation, der Entwickler von PageMaker für den Macintosh und den PC, erkannte die Notwendigkeit einer standardisierten Methode, digitale Daten auszutauschen, und entwickelte zusammen mit mehreren Herstellern das *Tagged Image File Format* (TIFF). TIFF unterstützt viele Eingabeeinheiten und Datenkomprimierungstechniken und bietet die Möglichkeit, Neuerungen in einer einfachen, kontrollierten Art hinzuzufügen.

TIFF unterstützt Scanner, Zeichenprogramme und Kameras (*Bild 1*). Es arbeitet genausogut mit einer einfachen Windows Paint-Zeichnung, wie mit einem komplizierten medizinischen Scanner. TIFF-Dateien enthalten Kennungen (tags), die nicht nur die Höhe und Breite des Bildes beschreiben, sondern enthalten auch Auflösungsinformationen, Informationen über die Graustufen oder Farben und das Komprimierungsverfahren, mit dem sie erstellt wurden.

Augenblicklich unterstützt TIFF eine modifizierte Version der RL-Komprimierung CCITT Gruppe 3. CCITT/3 bietet eine vernünftige Komprimierung für Schwarz/Weiß-Bilder mit 200 bis 300 dpi, indem es kontinuierliche schwarze oder weiße Bildpunkte erkennt. Diese werden durch ein eindeutiges Codewort ersetzt, womit die Originaldaten wiedergewonnen werden können. Die Komprimierung kommt zustande, da die Codeworte typischerweise kürzer sind als die Daten, die ersetzt werden. Die CCITT/3-Komprimierung für Grauwerte oder Farbdarstellungen ist nicht so gut, da hier nicht so viele redundante Informationen enthalten sind, wie in Schwarz/Weiß-Bildern. Aus diesem Grund unterstützt TIFF auch ein einfaches gepacktes Format, bei dem Daten so dicht wie möglich in Bytes untergebracht werden, wobei keine Bits überflüssig sind, außer am Ende einer Reihe. Spezielle Komprimierungsverfahren für Grau und Farbbilder werden noch kommen. Microsoft, der derzeitige Koordinator der TIFF-Spezifikation, wird Ihre Ratschläge begrüßen.

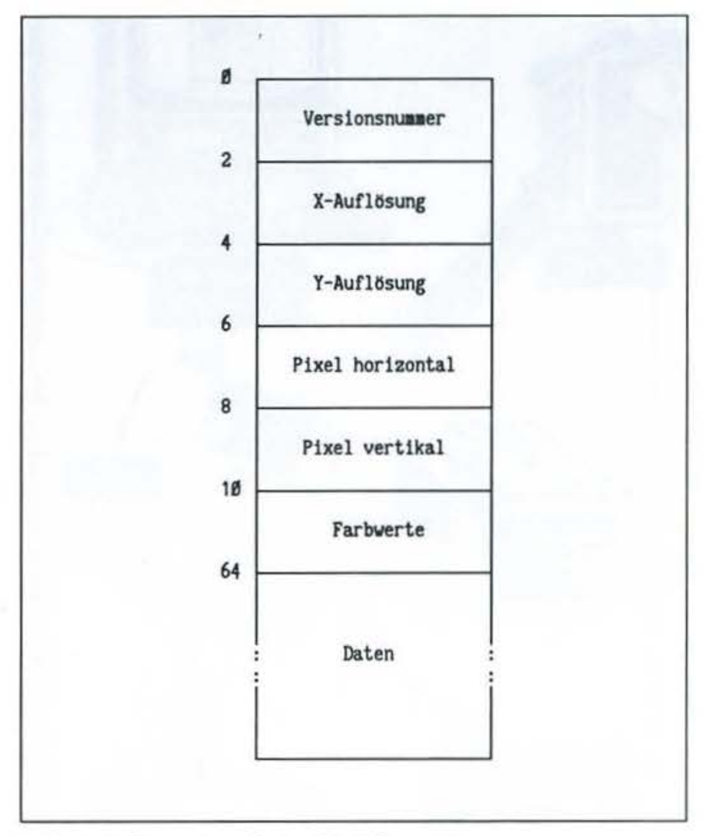


Bild 2: *Aufbau eines festen Dateiformats.*

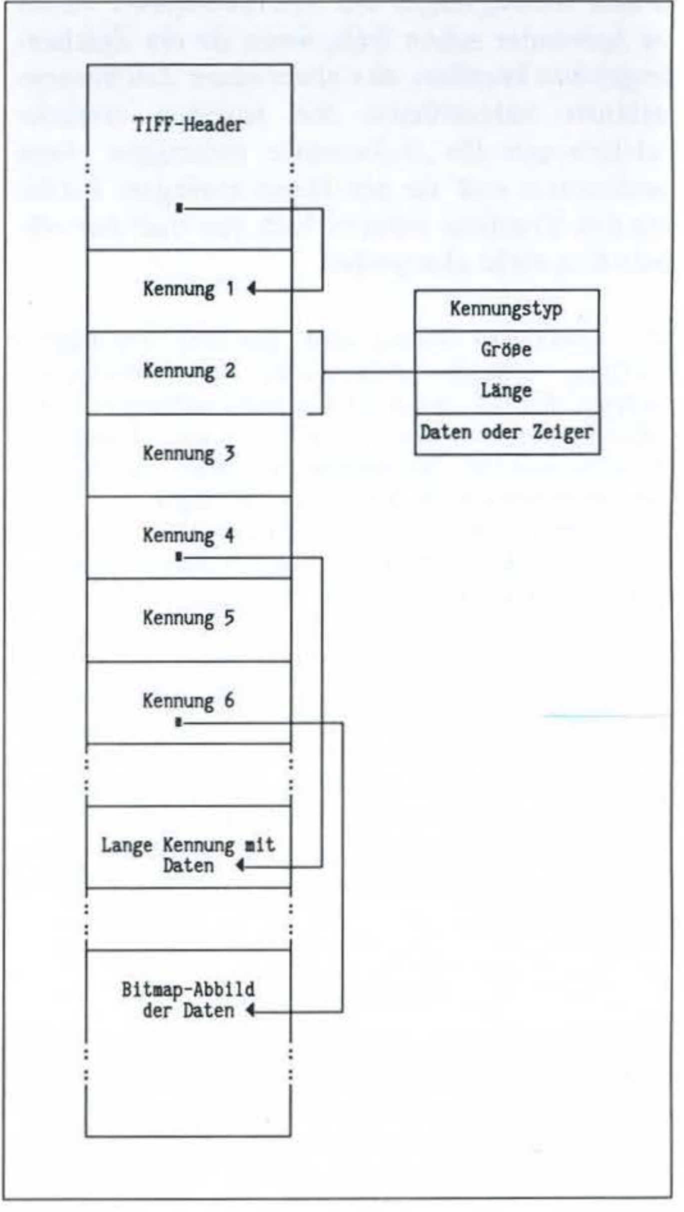


Bild 3: *TIFF verwendet Informationen mit Kennungen, eine Methode die vornehmlich bei Datenbanken zu finden ist. Kennungen definieren die Attribute des Bildes und wie das wirkliche Bild gespeichert ist. Jedes Bild besitzt einen Header, an den sich Kennungen anschließen, die Informationen beschreiben, die irgendwo anders in der Datei enthalten sind.*

TIFF ermöglicht es, Neuerungen hinzuzufügen, ohne daß bestehende Software jedesmal umgeschrieben werden muß. Will zum Beispiel eine Applikation eine Kennung »Datum der Entstehung« zum Bild hinzufügen, kann das beim Programmieren der Applikation geschehen. Andere Applikationen, die die TIFF-Dateien lesen, brauchen nicht geändert zu werden, außer wenn sie diese neue Datumskennung auswerten wollen. Sie können in den meisten Fällen ohne Probleme die unbekannten Kennungen ignorieren. Die TIFF-Spezifikation versichert, daß der Strukturierung der Daten sehr viel Aufmerksamkeit gewidmet wurde, um zukünftige Erweiterungen leicht implementieren zu können. Sollten in Zukunft Farbscanner verfügbar sein, so wird TIFF auch diese unterstützen können.

Die Arbeitsweise von TIFF

Herkömmliche Zeichenprogramme wie PC Paintbrush benützen eine feste Dateiorganisation (*Bild 2*). Die Organisation und der Platz jedes Wertes ist vor dem Lesen bekannt. Die Auflösung in der X-Achse zum Beispiel steht immer in den Bytes 2 und 3 des Satzes.

TIFF ist nicht positionsabhängig, stattdessen benutzt es Informationen mit Kennungen, eine Methode die vornehmlich bei Datenbanken zu finden ist. Kennungen definieren die Attribute des Bildes und wie das wirkliche Bild gespeichert ist. Jedes Bild besitzt einen Header, an den sich Kennungen anschließen, die Informationen beschreiben, die irgendwo in der Datei enthalten sind. Kennungen beinhalten den Namen der Kennungs-Information, die Größe und Länge der in der Kennung beschriebenen Information, und einen Zeiger auf die wirkliche Information (*Bild 3*). Kennungen können um alle zum Speichern des Bildes benötigten Informationen erweitert werden. Sie können deshalb so viele Kennungen anfügen, wie Sie benötigen. Für einfache Bilder brauchen Sie nur einige Kennungen, für komplexe hingegen können es sehr viele sein.

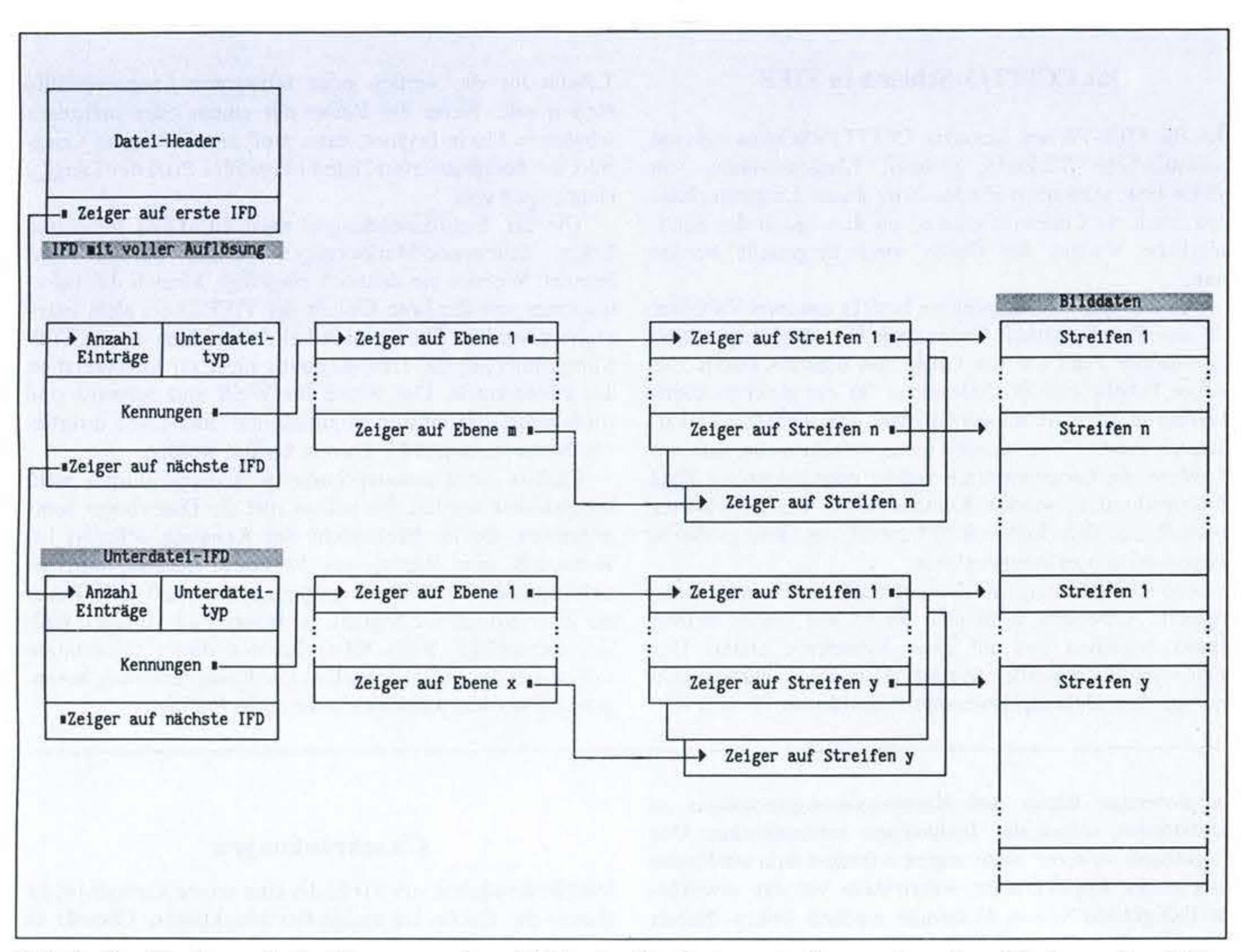


Bild 4: *Der Header und die Kennungen einer Bilddatei werden Image File Directory (IFD) genannt. Die Struktur erlaubt es auch, mehrere Versionen eines Bildes in derselben Datei zu speichern.*

Der Header und die Kennungen einer Bilddatei werden Image File Directory (IFD) genannt. Die Struktur erlaubt es auch, mehrere Versionen des Bildes in derselben Datei zu speichern. Dies kann sehr hilfreich sein, wenn Sie eine Version mit voller Auflösung zum Bearbeiten und Drucken und eine Version mit niedrigerer Auflösung für eine höhere Bildschirm-Anzeigegeschwindigkeit in einer Datei speichern wollen, wie dies in *Bild 4* gezeigt ist. Jede Version hat ihre eigene IFD mit eindeutigen Attributen und Zeigern zu den aktuellen Daten. TIFF erlaubt Ihnen das Speichern in Streifen mit variabler Länge oder in Bildpunktzeilen, um Ihnen wahlfreien Zugriff zu jedem Teil des Bildes zu ermöglichen. Brauchen Sie nur einen Teil eines sehr großen Bildes und haben dieses in Streifen abgespeichert, können Sie nur diejenigen Streifen expandieren und laden, die Sie benötigen. Beim Verändern eines Bildes ist es ebenfalls nur notwendig, die veränderten Streifen des Bildes zu aktualisieren, nicht das ganze Bild.

Vorteile

Für Software- und Hardware-Entwickler sind die Vorteile eines Standardformats für den Austausch digitaler Daten groß. Statt der Unterstützung zahlreicher Formate für jeden Scannerhersteller, jeder Dektop-Publishing-Applikation und jedes Zeichenprogramms, brauchen die Hersteller nur ein Dateiformat zu betreuen. Sogar die Unterstützung eines komplizierten Formats, wie TIFF es verwendet, ist leichter, als die ständig zunehmende Anzahl verschiedener Formate.

TIFF ist ein »großzügiges« Dateiformat. Statt der Rückführung aller Bilder auf den kleinsten gemeinsamen Nenner, ist TIFF für die Bearbeitung verschiedener Arten von Bildern, wie Strichvorlagen, Graustufen, Farben und auch verschiedene Auflösungen, ausgelegt. TIFF unterstützt verschiedene Komprimierungsverfahren und ist sogar so flexibel, daß der Komprimierungs-Algorithmus an den Datentyp angepaßt werden kann. TIFF wird erweitert werden, um

Das CCITT/3-Schema in TIFF

Das im TIFF-Format benutzte CCITT/3-Schema erkennt kontinuierliche Verläufe, genannt Längenverläufe, von weißen bzw. schwarzen Pixeln. Jeder dieser Längenverläufe wird durch ein Codewort ersetzt, mit dem später der kontinuierliche Verlauf der Daten wiederhergestellt werden kann.

Das Komprimierungsschema besteht aus zwei Tabellen. Die eine Tabelle enthält Codewörter für verschiedene Verläufe weißer Pixel von der Länge von 0 bis 63 Pixeln. Die andere Tabelle enthält Codewörter für die gleichen kontinuierlichen Verläufe schwarzer Pixel. Die Tabellen enthalten auch Werte für vereinzelte Längenverläufe oberhalb von 63. Wenn die Längenverläufe weißer oder schwarzer Pixel 63 überschreiten, werden Kombinationen der Codewörter oberhalb und unterhalb von 63 benutzt, um diese größeren Längenverläufe zu komprimieren.

Jede Reihe Bitmaps wird unabhängig voneinander komprimiert. Außerdem muß jede Reihe mit einem weißen Muster beginnen und auf einer Bytegrenze enden. Der Beginn mit einem weißen Muster liefert die Synchronisation und sagt dem Dekomprimierungs-Algorithmus, ob er in der Tabelle für die weißen oder schwarzen Längenverläufe suchen soll. Wenn die Reihe mit einem oder mehreren schwarzen Pixeln beginnt, dann muß am Anfang im Codewort der komprimierten Daten ein weißes Pixel der Länge 0 eingetragen sein.

Die für Fax-Anwendungen nach CCITT/3 gebräuchlichen Zeilenende-Markierungen (EOL) werden nicht benutzt. Werden sie dennoch eingefügt, können die Informationen von der Lese-Einheit der TIFF-Datei nicht interpretiert werden. Ebenso paßt bei Anwendung der CCITT-Komprimierung die TIFF-Kennung nicht zur Interpretation der Photometrie. Die Werte für Weiß und Schwarz sind vordefiniert. Sie müssen eventuell alle Binärdaten invertieren, bevor sie in CCITT-Format kodiert werden.

Füllbits einer unkomprimierten Bildzeile dürfen nicht komprimiert werden. Sie sollten nur die Datenlänge komprimieren, die im Breitenfeld der Kennung definiert ist. Wenn z.B. eine Bitmap aus 10 Pixeln besteht, wird sie unkomprimiert in 2 Bytes gespeichert, so daß jede Reihe mit einer Bytegrenze beginnt. In diesem Fall würden 6 Füllbits hinzugefügt. Beim Komprimieren dieser Information sollten nur die ersten zehn Bits zur Komprimierung herangezogen werden, jedoch nicht die sechs Füllbits.

---

kompliziertere Bilder und Komprimierungstechniken zu unterstützen, sofern die Technologie voranschreitet. Das Hinzufügen weiterer Neuerungen erfordert kein Änderung bestehender Applikationen, sofern diese von den erweiterten Fähigkeiten keinen Gebrauch machen wollen. Neben der Behandlung komplizierter Bilder arbeitet TIFF auch mit einfachen Bildern, zum Beispiel von Zeichenprogrammen, da die Applikation auch nur diejenigen Teile von TIFF verwenden kann, die sie benötigt.

Ein weiterer Vorteil von TIFF ist die Maschinenunabhängigkeit. Sie können es mit den Mikroprozessoren von Intel oder Motorola, den Betriebssystemen UNIX, MS-DOS oder dem des Macintosh verwenden, was Mitch Murphy, den Projektmanager von OptiGraphics veranlaßte, auf TIFF umzustellen. OptiGraphics, die Workstations für Ingenieure und Scanner für große Dokumente herstellen, suchten ein Dateiformat für PCs und UNIX-Computer. Murphy wählte TIFF, um seine Applikationen verbessern zu können und kompatibel zu bleiben. Produkte der Gruppe von Murphy, wie View! und Markup!, sind Microsoft Windows-Applikationen, mit denen Sie große Ingenieur- und Architekten-Zeichnungen betrachten, und kleinere Änderungen vornehmen können. Der Scanner gibt die Bilder in TIFF aus, und View! bzw. Markup! lesen die TIFF-Ausgabe und zeigen sie an. Benötigt eine Zeichnung intensivere Bearbeitung, gibt es Produkte, die eine Tiff-Datei automatisch mit Vektoren versehen, um sie auf einem CAD-System verwenden zu können.

Einschränkungen

Die Großzügigkeit von TIFF, die eine seiner Vorteile ist, ist ebenso die Quelle für einige Beschränkungen. Obwohl es für einen Scanner nicht schwer ist, eine TIFF-Bilddatei zu schreiben, ist es schwierig einen TIFF-Leser zu schreiben. Der Präsident von MacroMinds, Marc Cantor, betrachtet es als Fehler von Aldus und Microsoft, daß die TIFF-Spezifikation keinen Dekodieralgorithmus enthält. Derzeit muß jede Gesellschaft ihren eigenen Parser programmieren. Cantors Firma, deren Programm Graphic Works ein gescanntes TIFF-Dokument jeder Größe liest, hat ihren eigenen Parser geschrieben und bietet diesen auch zum Verkauf an. Um das Problem zukünftig zu beheben, wird Microsoft TIFF-Werkzeuge anbieten.

Eine weitere Einschränkung ist, daß es keine Standard-TIFF-Datei gibt, das heißt, keine gemeinsame Untermenge von Eigenschaften, die jeder, der den Namen TIFF verwendet, unterstützen muß. Verschiedene Hersteller können verschiedene Kennungen von TIFF unterstützen. Murphy von OptiGraphics meint, daß TIFF so mächtig ist, daß mehrere Dialekte entstehen werden. Um dies zu vermeiden, schlagen er und andere vor, gemeinsame Werkzeuge zu verwenden, die zwischenzeitlich vermehrt angeboten werden. Murphy hat eine Bibliothek von Funktionen, um auf TIFF-Dateien in einer geordneten Weise zuzugreifen. Sie können bei ihm kostenlos bezogen werden (Telefon 001-

612-292-6060), und von Ihnen auf Ihre eigene Verantwortung eingesetzt werden. Die Entwicklungsingenieure bei Hewlett-Packard haben einen entscheidenden Beitrag zur TIFF-Spezifikation geleistet, und ein »Guide to Tagged Image File Format« verfaßt. Diese Spezifikation ist auch Bestandteil des Windows Software Development Kit 2.0. Hewlett-Packard verfügt ebenso über Quellcode für das Schreiben, Lesen, Komprimieren, Expandieren und Debuggen von TIFF-Dateien. Zur Information können Sie sich an HPs ISV Marketing Department unter der Telefonnummer 001-303-350-4000 wenden. Dest Corporation, der Hersteller von Scannerhardware verfügt ebenso über eine Bibliothek von Funktionen, die er Herstellern von Desktop Publishing Systemen kostenlos zur Verfügung stellen will. Die Telefonnummer ist 001-408-946-7100. Tim Davenport von Aldus wird Ihnen einfache TIFF Dateien schicken, wenn Sie ihn unter der Nummer 001-206-622-5500 anrufen.

Der neue Standard?

TIFF wird sicherlich der Standard für den Umgang mit pixelorientierten Bildern. Firmen wie HP, Aldus und Opti-Graphics haben eine Menge Entwicklungsarbeit in TIFF gesteckt. Murphy meint, da sie von TIFF überzeugt sind, wollen sie andere auch überzeugen und bieten auch Unterstützung. David Rosenbloom von Cygnet Technologies, eine High End Telekommunikationsfirma, wählte für sein FAX-Produkt TIFF statt eines gewöhnlichen Formats, da es alles bietet, was sie brauchen: einen universellen Header mit einer unbeschränkten Anzahl von Feldern, und die Aussicht, der akzeptierte Standard zu werden. Andere Firmen, die angekündigt haben, daß sie TIFF unterstützen werden, sind Dest, DataCopy, Microsoft, Microtek, Media Cybernetics, New Image Technology und Software Publishing. Die Unterstützung von TIFF überbrückt die traditionellen Schluchten zwischen den PC- und den Macintosh-Lagern.

Obwohl Steve Carlson von Aldus der Initiator der TIFF-Spezifikation ist, ist jetzt Microsoft zuständig. Mit Microsoft anstelle eines Herstellers von Hardware oder Desktop Publishing Systemen, steht TIFF unter neutraler Koordination. Die Firmen sehen TIFF mehr als Standard, als eine Hilfe für Konkurrenten. Microsoft arbeitet derzeit mit denjenigen Firmen, deren Produkte Bilder scannen und archivieren. Man will aber auch mit den Herstellern von Seitenbeschreibungen, wie Adobe, DDL und Interpress zusammenarbeiten, um sicherzustellen, daß die von TIFF unterstützten Funktionen von zukünftigen Seitenbeschreibungssprachen auch unterstützt werden. Wenn es zum Beispiel kompatible Komprimierungsverfahren gibt, wird sich die Druckzeit erheblich verkürzen. Manny Vellon von der Microsoft Windows Marketing Group ist der neue TIFF-Administrator. Sie können bei ihm eine Kopie der TIFF-Spezifikation anfordern und ihm auch Ihre Vorschläge übersenden.

Derzeit ist der Desktop Publishing Markt am meisten an TIFF interessiert, aber er expandiert in Richtung OCR, FAX, dreidimensionaler CAD und Medizintechnik. Eines Tages werden Sie statt in die Bücherei zu gehen, um einen Kunstdruckband zu holen, von einer CD-ROM qualitativ hochwertige Bilder im TIFF-Format bekommen. Bald werden Sie eine technische Zeichnung als FAX auf Ihrem PC empfangen, ein Programm zur Vektorisierung dieser Zeichnung verwenden, sie mit Ihrem CAD-Programm bearbeiten, die Spezifikationen, die diese Zeichnung in einem anderen FAX begleitet, in OCR umwandeln, diese mit Ihrem Textverarbeitungssystem bearbeiten und alles in FAX-Form zurücksenden können. Diese Hardware- und Software-Applikationen werden zusammenarbeiten, sobald es einen Standard für den Austausch von digitalen Daten gibt. TIFF ist auf dem besten Weg dazu.

Nancy Andrews/Stan Fry

Gibt es preiswerter Informationen,
als die Erfahrungen* von Fachleuten zu nutzen?

*z.B. unsere Fachübersetzung MS Windows Programmer`s Reference!

Weiterhin bieten wir:
- 5 tägige Schulungen MS Windows-Training and Practice (ab 6 Teilnehmer auch in Ihrer Firma)
- Windows Programmierung
- qualifizierte Fachübersetzungen Ihrer Software-Dokumentation in englisch, französisch und spanisch

Incl. MwSt. plus Versandkosten nur DM 286,—

pcd

pcd Pirwitz Computer Dokumentation GmbH - 2300 Kiel 1 - Eckernförder Straße 259 - Tel.: 0431/54 20 70

Mitteilungen .... Mitteilungen .... Mitteilungen

Jüngste Version des MS-OS/2-SDK wird mit Vorversion des Presentation Managers ausgeliefert

Das Software-Entwicklungs-Kit (SDK) für das neue Betriebssystem MS-OS/2 enthält einen kompletten Satz Dokumentationen zu MS-OS/2, eine vorläufige Version des Betriebssystems MS-OS/2 1.1 und die Entwicklungswerkzeuge, die notwendig sind, um Applikationssoftware für MS-OS/2 einschließlich Presentation Manager zu entwickeln und zu testen. Diese Ausgabe des MS-OS/2-SDK ist die dritte Aktualisierung, seit Microsoft das Software-Entwicklungs-Kit im Juni 1987 zum ersten Mal ausgeliefert hat. Microsoft plant darüber hinaus vor Erscheinen der endgültigen Version von MS-OS/2, Version 1.1, weitere Aktualisierungen des SDK.

»Dieses MS-OS/2-Entwicklungs-Kit ist ein Meilenstein für Microsoft«, so Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH. »Die Verfügbarkeit des MS-OS/2-SDK für das OS/2-Betriebssystem mit Presentation Manager zeigt, daß wir unser Ziel erreicht haben, den Entwicklern in einem frühen Stadium die Basis zur Entwicklung von Applikationssoftware für MS-OS/2, Version 1.1, zur Verfügung zu stellen. Obwohl diese Vorversion naturgemäß noch einige Fehler enthält, sind wir überzeugt, daß die Entwickler von MS-OS/2-Applikationssoftware damit gute Fortschritte in der Entwicklung neuartiger und besonders leistungsfähiger Applikationssoftware machen. Mit dieser neuen SDK-Version beginnt der Countdown für die Entwicklung von MS-OS/2-Presentation Manager-Applikationen. Die meisten Softwarehäuser haben dieses grafische Interface als Entwicklungsumgebung gewählt. Deshalb ist das neue SDK ein Meilenstein für die Durchsetzung des neuen Standards.«

Das SDK enthält Dokumentationen mit mehr als 10.000 Seiten Umfang. Die Software wird sowohl auf 3,5-Zoll- wie auch auf 5,25-Zoll-Disketten geliefert. Dieser Satz besteht aus 63 Disketten. Die jüngste Version des MS-OS/2-SDK hat folgende Bestandteile:

- MS-OS/2, Version 1.1: die erste Vorversion des Betriebssystems MS-OS/2 mit Presentation Manager, der grafischen Bedieneroberfläche für das MS-OS/2-System.

- Das MS-OS/2-Programmierer-Toolkit: mit Entwicklungswerkzeugen, Utilities, Beispielprogrammen und Referenzmaterial.

- Die Microsoft Windows SDK-Version 2.03: jüngste Version des Microsoft Windows-Software-Entwicklungs-Kits zur Entwicklung von Applikationen unter Microsoft Windows 2.0 oder Microsoft Windows 386. Das Windows-Software-Entwicklungs-Kit enthält wertvolles zusätzliches Entwicklungs- und Instruktionsmaterial, das für die Presentation Manger-Umgebung wichtig ist.

- Microsoft-C und MASM, Version 5.1: die jüngste Version von Microsoft-C und des Microsoft Macro Assemblers für MS-OS/2, mit deren Hilfe Entwickler Applikationsprogramme für MS-OS/2 schreiben können und bei der Fehlersuche unterstützt werden.

Es besteht die kostenfreie Möglichkeit, Microsofts Online-Entwicklungs-Unterstützungsservice in Anspruch zu nehmen. Außerdem erhält der Kunde acht Video-Bänder mit jeweils zwei Stunden Laufzeit, die nützliche Informationen über die Entwicklung von MS-OS/2-Applikationssoftware enthalten. Alle MS-OS/2-SDK-Lizenznehmer erhalten jeweils kostenlose Aktualisierungen des SDK.

Was Sie noch über das Transportprotokoll des MS-OS/2 LAN-Managers wissen sollten

Die Microsoft GmbH veröffentlichte unlängst eine umfassende Strategie für eine Auswahl von Netzwerk-Transport-Protokollen mit austauschbaren Hardware-Treibern für den MS-OS/2-LAN-Manager, die Netzwerk-Software des neuen Betriebssystems MS-OS/2. Die Transportprotokolle sind eine Implementation des IBM-Netbeui, ISO-Transport-Klasse-4 sowie TCP/IP und werden sowohl für MS-DOS als auch für MS-OS/2 lieferbar sein. Paul Maritz, General Manager des Microsoft Netzwerk-Bereichs, erläuterte hierzu auf der ersten *Advanced Network Development Conference* in San Francisco: »Unsere Strategie zielt auf die Möglichkeit der Zusammenarbeit verschiedener MS-OS/2-Netzwerke durch Konzentration auf eine geringe Anzahl von Standardprotokollen und austauschbaren Hardware-Treibern. Mit Unterstützung unserer Partner bei der Entwicklung von Netzwerk-Software konzentrieren wir unsere Entwicklungsanstrengungen und unsere Unterstützung auf diese Standard-Transportprotokolle.«

Die Datenübertragung erfolgt unter Nutzung des Standard-MAC(Media Access Control)-Interfaces. Die Standard-Transportprotokolle sind für das MAC-Interface geschrieben, das von Microsoft in Zusammenarbeit mit der 3Com Corporation definiert wurde. Das Standard-MAC-Interface bietet eine einfache Möglichkeit, Treiber für Netzwerk-Adapterkarten zu schreiben. Es erlaubt darüber hinaus dem Netzwerk, mit einem Mix von Transportprotokollen und Hardware zu arbeiten. So kann beispielsweise das Netbeui-Protokoll sowohl auf einem Ethernet- als auch auf einem Token-Ring-Netzwerk laufen. Als Teil der Zusammenarbeit zwischen Microsoft und 3Com unterstützen die beiden Unternehmen die Entwicklungen von Transportprotokollen für Netbeui- und ISO-Datenübertragungen. In Zusammenarbeit mit den Unternehmen Madge Networks Limited und Retix bietet die 3Com Corporation zwei Standard-Datenübertragungs-Produkte an. Madge Networks entwickelt das IBM-Netbeui-kompatible Transportprotokoll, das auch den MS-OS/2-LAN-Manager in die Lage versetzt, mit IBM-PC Netzwerk-Produkten zusam-

Mitteilungen .... Mitteilungen .... Mitteilungen

menzuarbeiten. 3Com entwickelt die ISO-Datenübertragungs-Software, basierend auf einem Programm, das von Retix stammt. 3Com wird die ISO-Transport-Software gemäß der »Standard NetBIOS-to-ISO Mapping Specification« implementieren, wie sie vom TOP/NetBIOS-Architektur-Komitee festgelegt wurde.

Eric Benhamou, Vizepräsident der Software Products Division von 3Com meint: »Der MS-OS/2-LAN-Manager ist das Kernstück in der 3Com-'3+Open'-Produktlinie. Die Verfügbarkeit von Standard-Transportprotokollen ist der wesentliche Punkt, um den Anforderungen unserer Kunden zu entsprechen«.

Der Präsident der Firma Retix, Andy De Mari, führt aus: »Wir liefern die Technologie, auf deren Basis 3Com die ISO-Transport-Software entwickelt. Das ISO-Netzwerk-Protokoll wird eine größer werdende strategische Bedeutung haben, weil die Anwender Unterstützung für Standardprotokolle fordern.«

Robert Madge, Gründer der Madge Networks Limited: »Natürlich wird der MS-OS/2-LAN-Manager durch die Unterstützung für das IBM-Netbeui-Transportprotokoll verbessert. Die Kompatibilität zu dem PC-LAN-System von IBM wird für die Netzwerk-Anwender wesentlich sein«.

TCP/IP wird von der Excelan Inc. geliefert. Diese Firma entwickelt ein TCP/IP-Transportprotokoll für den MS-OS/2-LAN-Manager. Kanwal Rekhi, Executive Vice President of Business Development der Excelan Inc., die in diesen Tagen bekanntgab, daß sie ebenfalls einen Lizenzvertrag für den MS-OS/2-LAN-Manager unterschrieben hat, dazu: »TCP/IP ist ein wichtiges LAN-Transportprotokoll, das heute schon eine effektive Möglichkeit der Datenübertragung in einer breiten Palette von Umgebungen bietet. Wir freuen uns, eng mit Microsoft zusammenzuarbeiten, um TCP/IP für den MS-OS/2-LAN-Manager verfügbar zu machen«.

Die Implementation von TCP/IP erfolgt gemäß den *Standard NetBIOS-to-TCP Mapping Specification*, enthalten in RFC1002 des *Internet Activities Board*.

Microsoft wird Standardversionen dieser drei Transportprotokolle an seine LAN-Manager-OEM-Kunden liefern. IBM-Netbeui wird im zweiten Quartal 1988 verfügbar sein. ISO und TCP/IP stehen mit Beginn des vierten Quartals 1988 zur Verfügung. OEM-Kunden, die eine spezielle Version für die TCP-, IBM-Netbeui- oder ISO-Protokolle haben möchten, um zum Beispiel intelligente Netzwerk-Adapter zu betreiben, werden von Microsoft mit den Firmen Excelan, Madge oder Retix zusammengebracht.

Microsoft liefert zusätzlich an seine OEM-Kunden eine MAC-Treiber-Bibliothek für eine Reihe von Netzwerk-Adapterkarten, einschließlich 3COM-Etherlink und IBM-Token-Ring.

Das Netzwerk-Transportprotokoll ist verantwortlich für die Datenübertragung innerhalb des lokalen Netzwerkes und umfaßt Software-Funktionen von der NetBIOS-Ebene bis hinunter zu den Schnittstellen für die Netzwerk-Adapterkarten. Es ist als ein Satz von Gerätetreibern implementiert. Unter Microsofts DOS-basierender Netzwerk-Software, MS-Net, bietet jeder OEM-Lieferant seine eigenen Transportprotokolle an. Die OEM-Lieferanten werden auch weiterhin diese Option beim MS-OS/2-LAN-Manager haben, sie können sich nun aber auch dafür entscheiden, die Vorteile von Standard-Transportprotokollen zu nutzen. Dieses Konzept schließt nicht den Einsatz von anderen Transportprotokollen, wie DECNET oder XNS unter dem MS-OS/2-LAN-Manager aus. So plant zum Beispiel 3Com die Unterstützung von XNS-Protokollen auf seinen Etherlink-, Token-Ring- und IBM-Token-Ring-Karten als Wachstumsfahrt für »3+R«-Anwender an.

Führende Software-Unternehmen unterstützen MS-OS/2-Netzwerkplattform

Auf der ersten »Microsoft Advanced Network Development Conference« in San Francisco haben mehr als 35 führende Software-Entwicklungsunternehmen angekündigt, daß sie mit mehr als 50 Anwendungsprogrammen den Microsoft OS/2-LAN-Manager, die LAN-Software für das neue Betriebssystem MS-OS/2, sowie den SQL-Server, ein Hochleistungs-Datenbank-Verwaltungssystem für MS-OS/2-Netzwerk-Server, unterstützen. Darüber hinaus haben weitere 29 führende Hardware-Hersteller bestätigt, daß sie Lizenzen für den Microsoft OS/2-LAN-Manager erwerben wollen.

Einschließlich der bislang vergebenen Lizenzen sind nun 35 namhafte Unternehmen der Computerbranche Lizenznehmer für den MS-OS/2-LAN-Manager. Darunter finden sich Namen wie AT&T, NCR Corporation, Tandy Corporation, Ungermann Bass, Nokia Data, Olivetti, 3Com Corporation, Hewlett-Packard, Excelan Inc., Digital Communications Associates Inc. und viele andere. IBM setzt darüber hinaus die Microsoft-OS/2-LAN-Technologie bei eigenen OS/2-LAN-Servern ein.

Der MS-OS/2-LAN-Manager unterstützt drei Arten von Anwendungen:

- Standard-MS-DOS- und OS/2-Anwendungen, die transparent in einem Netzwerk arbeiten,
- Anwendungen, die speziell das MS-OS/2-LAN Anwendungs-Programm-Interface (APIs) nutzen,
- Anwendungen, die den SQL-Server-Interfaces ansprechen und Daten vom SQL-Server aufrufen.

Beim ersten Typ, den Standard-MS-DOS und MS-OS/2-Anwendungen, erweitert der MS-OS/2-LAN-Manager die File-Struktur und die Geräte-Schnittstellen auf das gesamte Netzwerk, so daß MS-OS/2-Anwendungen im gesamten Netz transparent unterstützt werden. Diese transparente Unterstützung bezieht sich auch auf MS-DOS-Arbeitsplatz-Computer und MS-DOS-Anwendungen, einschließlich Microsoft Windows-Anwendungen.

Mitteilungen .... Mitteilungen .... Mitteilungen

Der zweite Typ von Anwendungen ist speziell für den MS-OS/2-LAN-Manager ausgelegt. Der MS-OS/2-LAN-Manager bietet Anwendungs-Programm-Schnittstellen (APIs), die es Programmen erlauben, Netzwerk-Operationen und Netzwerk-Ressourcen zu steuern bzw. in Anspruch zu nehmen. Anwendungen, die über diese Schnittstellen laufen, sind für die MS-OS/2-LAN-Manager-Umgebung optimiert.

Der dritte Typ von MS-OS/2-LAN-Manager-Anwendungen sind solche, die als *Frontend* auf dem PC laufen und den SQL-Server unterstützen, oder die als *Backend* ein Datenbank-Verwaltungssystem darstellen.

Der SQL-Server ist für OS/2-Netzwerke optimiert und arbeitet seinerseits mit den LAN-Manager Anwendungs-Programm-Schnittstellen. Der SQL-Server kann beispielsweise als ein MS-OS/2-LAN-Manager-Service installiert werden, wobei er die SQL-Server-Administration vereinfacht.

MS-OS/2-LAN-Manager-Anwendungen sind bereits von führenden Software-Herstellern angekündigt worden. Consumers Software Inc., ein maßgebender Lieferant von Software für *elektronische Post*, kündigte mit »MS-OS/2 Network Courier« ein *intelligentes* Programm an, das MS-DOS-, Microsoft Windows- und MS-OS/2-Arbeitsplatz-Computer unterstützt, die mit dem MS-OS/2-LAN-Manager arbeiten. *Network Courier* wurde für das NetBIOS-Interface geschrieben, welches durch den MS-OS/2-LAN-Manager unterstützt wird. Network Courier unterstützt darüber hinaus andere MS-OS/2-LAN-Manager-APIs, wie *Security Facilities* und *Named Pipes*.

Als eines der führenden Software-Häuser für Buchhaltungs-Software hat TLB Inc. eine neue Version seines Paketes *Solomon III* angekündigt. Dieses Buchhaltungsprogramm wird mit einem Datenbank-Server arbeiten, um von den Anwendern gemeinsam benutzte Daten zu speichern und zur Verfügung zu stellen sowie Sicherheitsvorkehrungen, wie das Sperren von Datensätzen, zu gewährleisten. Die Vorteile des MS-OS/2-LAN-Managers hinsichtlich des Einsatzes beim »Solomon III«-Paket liegen in der MS-OS/2-Multitasking-Fähigkeit und der Verfügbarkeit eines großen Speichers auf dem Server, wobei der MS-OS/2-LAN-Manager gleichzeitig eine nahtlose Verbindung mit den auf den Arbeitsplatz-Computern laufenden Anwendungsprogrammen schafft. Die Entwickler von Solomon III haben außerdem weitere interessante Möglichkeiten des MS-OS/2LAN-Managers, wie Named Pipes, Druckerfunktionen und Netzwerk-Verwaltungsfunktionen genutzt.

Für den SQL-Server sind eine Reihe von Arbeitsplatz-Computer-Anwendungen angekündigt worden. Der SQL-Server stellt einen erheblichen Fortschritt in der Datenbanktechnik dar, weil er Funktionen wie Speicherprozeduren, Trigger und einen hochentwickelten transaktionsorientierten SQL-Kern vereint. Diese Funktionen garantieren die Integrität der Daten und eine hohe Leistungsfähigkeit in einer LAN-Multiuser-Umgebung.

Auf der Advanced Network Development Conference wurden eine Reihe weiterer interessanter Anwendungen neben den vorab beschriebenen vorgestellt. Jede dieser Anwendungen machte deutlich, daß der MS-OS/2-LAN-Manager die richtige Wahl als Basis für lokale Netzwerke ist. Er bietet eine hochentwickelte, offene Plattform für die Entwicklung einer neuen Generation von Netzwerk-Anwendungen. Die folgende Liste stellt eine komplette Zusammenstellung der Produkte dar, die auf der Advanced Network Development Conference angekündigt bzw. vorgestellt worden sind. Neben den Produkten sind auch deren Hersteller aufgeführt. Die Aufstellung schließt nicht jene hunderte von MS-DOS-Anwendungsprogrammen ein, wie Lotus 1-2-3 und MS-Word, die heute auf MS-Net laufen und in gleicher Weise auf MS-OS/2-LAN-Manager-Netzwerken laufen werden.

OS/2-Anwendungsprogramme

Produkt	Hersteller
Platinum	Advanced Business Microsystems
Informix-4GL	Informix Software Inc.
Informix-SQL	Informix Software Inc.
Informix-ESQL/C	Informix Software Inc.
C-ISAM	MicroPro International
Rbase	Microrim Inc.
ORACLE für OS/2	Oracle Corporation
Bridge/386	Softbridge Microsystems Corp.
Bridge/286	Softbridge Microsystems Corp.
Bridge/OS	Softbridge Microsystems Corp.
WordPerfect	WordPerfect Corporation
PlanPerfect	WordPerfect Corporation
DataPerfect	WordPerfect Corporation
WordPerfect Office	WordPerfect Corporation
DBMS und Networking Software	WordTech Systems, Inc.

MS-OS/2-LAN-Manager-spezifische Anwendungsprogramme

Produkt	Hersteller
SuperProject Expert/2	Computer Associates International Inc.
ACCPAC Plus System Manager/2	Computer Associates International Inc.
CAPS, einschließlich Remote Connect und TFD II Fault Tolerance	Congent Data Technologies Inc.
Maxess SNA Gateway für OS/2	Communications Solutions Inc.
Higging Office Automation Software	Conetic Systems Inc.
Lanscope LAN Management System	Connect Computer

Mitteilungen .... Mitteilungen .... Mitteilungen

Produkt	Hersteller
The Network Courier	Consumers Software Inc.
DUET	Consumers Software Inc.
Select OS/2	
Communications Server	Digital Communications Associates
SQLBase	Gupta Technologies Inc.
PC/Focus-MultiUser	Information Builders Inc.
SmartWare	Informix Software Inc.
Lomax Utilities	J. A. Lomax Associates
LAN Shell	LAN Services Inc.
LANtrail	LAN Services Inc.
LANfolio	LAN Services Inc.
ReferencePoint	LAN Systems Inc.
MDBS III	Micro Data Base Systems Inc.
LANscape OS/2	Ncompass Software Inc.
netwise TOOLS	Netwise Inc.
Sniffer	Network General Corporation
sna62 Facility	Orion Network Systems Inc.
sna0123 Facility	Orion Network Systems Inc.
netPATH SNA-3270	Pathway Design Inc.
COAXGATE	Pathway Design Inc.
cc:Mail	PCC Systems
Add-A-Terminal	Softronics Inc.
Solomon III	TLB Inc.
Zukünftige Produkte	Microsoft Corporation

SQL-Server-Anwendungsprogramme

Produkt	Hersteller
Paradox	Ansa Software/Borland
dBASE IV	Ashton-Tate Corporation
Database Products	DateEase International Inc.
ACCESS/STAR	DB/ACCESS Inc.
PC/Focus-MultiUser	Information Builders Inc.
GURU	Micro Data Base Systems Inc.
MDBS III	Micro Data Base Systems Inc.
KnowledgeMan/2	Micro Data Base Systems Inc.
Workgroup Information Software	Saros Corporation
SQL Datenbank	Software Products International
Zukünftige Produkte	Symantec Corporation
Zukünftige Produkte	Microsoft Corporation

Vorabversion des MS-OS/2-LAN-Managers für viertausend MS-OS/2-Software-Entwicklungskit-Anwender angekündigt

Für mehr als viertausend Besitzer des Microsoft-Software-Development-Kits (SDK) kündigt Microsoft für das neue Betriebssystem MS-OS/2 die Auslieferung einer neuen Vorversion des Microsoft-OS/2-LAN-Managers an. Die SDK-Version des MS-OS/2-LAN-Managers arbeitet mit MS-OS/2, Version 1.0, und der Vorversion von MS-OS/2, Version 1.1, die im April an alle SDK-Besitzer ausgeliefert wurde. Der MS-OS/2-LAN-Manager wird im Laufe des Juni ausgeliefert.

Die Teilnehmer der Microsoft-Advanced-Network-Development-Konferenzen werden darüber hinaus eine Vorversion des MS-OS/2-LAN-Managers im Rahmen des MS-OS/2-LAN-Entwicklungspakets erhalten.

Der MS-OS/2-LAN-Manager ist bereits bei mehr als hundert OEM-Kunden und ausgesuchten Anwendern installiert und in Betrieb. Die jetzige Aktualisierung des Software-Entwicklungs-Kits für MS-OS/2 entspricht dem Ziel von Microsoft, Entwicklern so früh wie möglich die Basis für die Entwicklung und Prüfung von LAN-Manager-Applikationen zur Verfügung zu stellen. Das neueste Release des MS-OS/2-Entwicklungs-Kits ist die vierte Aktualisierung seit der ersten Auslieferung im Juni 1987 und wird Ende Juli 1988 verfügbar sein. Bevor die endgültige Version von MS-OS/2, Version 1.1, herauskommt, wird Microsoft noch weitere Aktualisierungen des Software-Entwicklungs-Kits ausliefern.

Die neue Version 1.5 von Microsoft Excel für den Macintosh bietet eine höhere Leistung und vereinfacht die Applikationsentwicklung

Microsoft hat die Auslieferung einer neuen Version von Microsoft Excel, dem führenden Tabellenkalkulations-Programm für den Apple-Macintosh, angekündigt. Die Version 1.5 von Microsoft Excel bietet Macintosh-Anwendern eine leistungsfähige und einfach zu handhabende Umgebung für die Entwicklung kundenspezifischer Applikationen. Die neue Version umfaßt nun Multitasking, Farbunterstützung für den Macintosh-II, erweiterte Möglichkeiten bei der Erstellung von Grafiken und 44 neue Arbeitsblatt-Funktionen.

Weil Tabellenkalkulations-Programme im Markt eine wichtige Rolle spielen, schreiben eine Reihe von unabhängigen Software-Herstellern Applikationen, die es ermöglichen, mit diesen Tabellenkalkulationen auf einfache Weise zu arbeiten. Dies trifft in besonderem Maße auch auf Microsoft Excel zu. Mit den neuen umfangreichen Möglichkeiten von Microsoft Excel 1.5 wird die Zahl der unabhängigen Entwickler von Applikationen weiter anwachsen, so die Stellungnahme von Christian Wedell, Geschäftsführer der deutschen Microsoft GmbH.

Die umfassenden Möglichkeiten von Microsoft Excel 1.5 stellen eine hervorragende Umgebung für die Entwicklung von Applikationen für vertikale Märkte und spezifische Funktionen, wie zum Beispiel Lagerbestandskontrolle, dar. Menüs, Befehle und Dialogfelder können bei Microsoft Excel 1.5 vollständig anwendungsspezifisch gestaltet werden, um speziellen Anforderungen gerecht zu werden. Der Anwender kann Applikationen mit einem maßgeschneiderten Interface entwickeln.

Mitteilungen .... Mitteilungen .... Mitteilungen

Excel 1.5 bietet nun Unterstützung des Multitasking des MultiFinder-Betriebssystems und der Farbmöglichkeiten des Macintosh-II. Es verwaltet nun bis zu 2.048 Datenpunkte (bisher 101) in einer Serie und enthält 30 neue Makrofunktionen und 44 neue Arbeitsblattformeln.

Die neue Excel-Version ist so ausgelegt, daß für den Macintosh- und den IBM-PC dieselbe Makro-Sprache und ein vergleichbares Interface benutzt werden. Microsoft Excel 1.5 läuft auf allen AppleShare-Netzwerken und ist kompatibel zur gesamten Macintosh-Anwendungsprogramm-Familie von Microsoft. Microsoft Excel läuft auf jedem Macintosh-Computer mit 512 Kbyte RAM, ein externer Massenspeicher von 800 Kbyte Größe wird empfohlen.

Registrierte Anwender von Microsoft Excel 1.0 erhalten die neue Version 1.5 zu den üblichen Update-Konditionen. Das Produkt wird ab Ende Juli 1988 in englisch und ab Ende August in Deutsch verfügbar sein.

Mit Microsoft-File 2.0 werden Datenbank-Applikationen auf dem Macintosh einfacher, schneller und flexibler

Microsoft hat eine wesentliche Erweiterung seines Datenbank-Verwaltungsprogramms Microsoft File für den Macintosh-Plus, -SE und -II angekündigt.

Die englische Version 2.0 arbeitet schneller, ist einfacher und darüber hinaus flexibler als frühere Versionen dieses Datenbank-Verwaltungsprogramms. Es wurden mehr als 120 Format-Vorlagen hinzugefügt, um den Einsatz von vorgedruckten Formularen zu vereinfachen. Eine erweiterte Kompatibilität zur Serienbrief-Funktion von Microsoft Word erleichtert das Erstellen von Formbriefen, Serienbriefen und Etiketten. Die neue Microsoft File-Version unterstützt darüber hinaus die Farbdarstellungs-Möglichkeiten des Macintosh-II, so daß Briefköpfe, Formulare und andere Ausdrucke nun farbig gestaltet werden können.

Neu hinzugekommen ist eine Diskette mit mehr als 120 vorformatierten Formularen, Etiketten und Bericht-Mustern, die den Anwender in die Lage versetzen, schnell zu der von ihm gewünschten Ausgabe zu gelangen. Um eine spezielle Ausgabe zu erzeugen, wählt der Anwender einfach das entsprechende Muster und gibt die Daten in das vordefinierte Formular ein. Microsoft-File 2.0 enthält darüber hinaus eine Hilfe-Liste sowie eine kontextbezogene Online-Hilfsfunktion. Das Makro-Paket »AutoMac« bietet eine einfache Automatisierung von Routine-Aufgaben und Berichten. File 2.0 unterstützt auch Sonder-Papierformate und Mehrspalten-Etiketten, so daß Etiketten und Adressenlisten einfach und schnell erstellt werden können.

Weitere Funktionen zur Verbesserung der Unterstützung bei der Herstellung von Formularen umfassen Memo-Felder für lange Textpassagen, Rechenunterstützung in Datenfeldern sowie einen anwenderdefinierbaren Fehlerwert für Feldgrenzen. Im Rahmen von Formatierungs-Reports kann der File-Anwender nun Fonts und Schriftarten für Absätze und ganze Texte definieren sowie Fonts und Größen für Überschriften und Fußnoten angeben.

Microsoft-File 2.0 läuft auf jedem AppleShare-Netzwerk und unterstützt die neuesten Drucker (auch Farbdrucker) die durch den Macintosh-II unterstützt werden.

Registrierte Anwender von Microsoft-File 1.0 können die Update-Regelung von Microsoft in Anspruch nehmen. Das Produkt wird Ende August 1988 in der englischen Version verfügbar sein.

Ergebnisse der Microsoft-Umfrage unter Softwareentwicklern

Im März dieses Jahres führte Microsoft ein Umfrage unter Softwareentwicklern durch. Gefragt wurde unter anderem nach der hauptsächlichen Verwendung von Programmiersprachen, Zielbetriebssystem für die Entwicklung, Kommunikationsmöglichkeiten, Vorschläge für die technische Unterstützung durch Microsoft und Arten der Weiterbildung. Etwa 1000 Fragebögen gingen ausgefüllt bei Microsoft ein. Der weitaus größte Teil (ca. 1/3) der Entwickler sitzt in Firmen mit eigener Entwicklung, zu den nächstgrößeren Gruppen gehören Softwarehäuser, Hochschulen und Unternehmensberater. Der weitaus größte Teil (ca. 60%) entwickelt Software für MS-DOS, an zweiter Stelle steht mit ca. 10% Microsoft Windows; OS/2 und andere Betriebssysteme (Xenix, Macintosh) liegen derzeit um die 5%. An Programmiersprachen werden am meisten Cobol, C, Fortran, BASIC und Pascal eingesetzt (in dieser Reihenfolge). Es werden alle wesentlichen Kommunikationseinrichtungen genutzt (BTX, Koppler, Modem, Datex-P), bevorzugt werden jedoch Modems. Für den Microsoft-Support wurden Erweiterungen in Richtung mehr Informationen auf Papier und Diskette, Anfragemöglichkeiten über Modem und Zugriff auf eine zentrale Support-Datenbank vorgeschlagen. Die meisten Entwickler sind durchaus bereit, für solche Verbesserungen auch zu zahlen.

Unter den Einsendern des Fragebogens wurden 10 Exemplare von Microsoft Excel verlost. Microsoft gratuliert den glücklichen Gewinnern:

- Detlev Bock, Petrikirchstr. 36, 3400 Göttingen
- GSE GmbH, Pixistr. 2, 8000 München 80
- Hermes Precisa International SA, Andrè Staehli, CH Yverdon-Les-Bains
- Otto F. Dittmar, Riedsaumstr. 22, 6700 Ludwigshafen
- ProCom GmbH, Claudia Geilenkirchen, Luisenstr. 41, 5100 Aachen
- Adapt Software/Hardware, Franz Reisinger, Staudgasse 7/2/9, A-1180 Wien
- Reinhold Werth, Sternstr. 17, 8000 München 22
- Dr. R. Herzig, 91, Rue de la Servette, CH-1202 Genf
- Jean-Pierre Huber, 19, Rue Maunoir, CH-1207 Genf
- Polymot GmbH, Zürcherstr. 23, CH-8640 Rapperswil

Die Grafikprogrammierschnittstelle von OS/2:

GPI: Eine Einführung in Präsentationsbereiche

Programmierer, die bereits Erfahrung mit Microsoft Windows haben, werden feststellen, daß der größte Teil des OS/2 Presentation Managers bereits bekanntes Gebiet ist. Die Bereiche Fensterverwaltung und Benutzerschnittstelle des Presentation Managers stammen ganz offensichtlich aus Windows und viele Fenstermeldungen sind identisch.

Die Grafikprogrammierschnittstelle (Graphics Programming Interface, GPI) ist jedoch im wesentlichen neu und bietet einige sehr wichtige Erweiterungen gegenüber dem Graphics Device Interface (GDI) von Windows. GPI stammt im wesentlichen von IBMs Graphical Data Display Manager (GDDM) und vom 3270 Graphics Control Program (GCP) ab, wobei einige Fähigkeiten wie Bitmaps und Regionen aus Windows stammen. In GPI findet sich auch der Einfluß vieler anderer Grafikschnittstellen, vom Graphical Kernel System (GKS) bis PostScript.

Eine der ersten Hürden bei der Annäherung an GPI ist das Verständnis des Konzepts des Präsentationsbereichs (presentation space oder kurz PS). Was es so schwierig macht ist, daß der Presentation Manager drei verschiedene Arten von Präsentationsbereichen unterstützt, den »Cached Mikro-PS,« den »Mikro-PS,« und den »normalen PS«.

Welche Art von Präsentationsbereich Sie für Ihre Anwendung wählen, hängt von zahlreichen Faktoren ab, von denen einige hier beschrieben werden sollen.

Das Thema wird noch interessanter, wenn Sie berücksichtigen, daß die Art des Präsentationsbereichs zu einem gewissen Grad mitbestimmt, wie eine Presentation Manager-Anwendung zum Zeichnen strukturiert wird. Ich werde dies anhand von drei Programmen demonstrieren, CACHEDPS, MICROPS und NORMALPS, wobei jedes einzelne eine der drei vorhandenen Arten von Präsentationsbereichen verwendet.

Jedes Programm zeigt die einfache Grafik an, die in *Bild 1* zu sehen ist. Der String »Graphics Programming Interface« wird in der Mitte des Fensters gezeichnet mit gepunkteten Linien von den Ecken des Strings zu den Ecken des Fensters. Der Text und die gepunkteten Linien werden rot angezeigt.

Die Zeichenfunktionen des GPI

Bevor ich näher auf die Präsentationsbereiche eingehe, wollen wir einen schnellen Blick auf die Funktionen werfen, die diese drei Programme zum Zeichnen des Textes und der gepunkteten Linien verwenden. Das Programm `CACHEDPS.C` ruft die GPI-Zeichenfunktionen in seiner Funktion `ClientWndProc` bei der Bearbeitung der Meldung `WM_PAINT` auf.



Bild 1: *Das Programm CACHEDPS unter dem OS/2 Presentation Manager.*

Viele der GPI-Zeichenfunktionen benötigen einen Punkt aus X- und Y-Koordinaten, der der Funktion als eine Struktur vom Typ `POINTL` übergeben wird. Die Struktur `POINTL` wird in der Header-Datei `OS2DEF.H` etwa so definiert:

```
typedef struct _POINTL
     {
     LONG x ;
     LONG y ;
     }
     POINTL ;
```

Der Datentyp `LONG` ist einfach eine 32-Bit unsigned long Ganzzahl. Um eine `POINTL`-Struktur zu definieren, können sie so vorgehen:

```
struct _POINTL ptl ;
```

Oder noch einfacher:

```
POINTL ptl ;
```

Gewöhnlich beginnen `POINTL`-Variablen mit dem Präfix `ptl`.

(Ein großer Unterschied zwischen GPI und anderen höheren Grafiksprachen ist, daß GPI keine Fließkommazahlen verwendet. Dies wurde aus Leistungsgründen so gemacht – Fließkommaberechnungen auf PCs sind immer noch nicht schnell genug für ausgesprochen interaktive grafische Systeme wie den Presentation Manager.)

GPI verwendet das Konzept der aktuellen Position. Um eine Linie zu zeichnen, besteht der erste Schritt darin, die aktuelle Position auf den Anfang der Linie einzustellen. Dazu müssen die Koordinatenwerte den X- und Y-Feldern der Struktur `POINTL` zugewiesen und dann die Funktion `GpiMove` aufgerufen werden:

```
ptl.x = ... ;
ptl.y = ... ;

GpiMove (hps,&ptl) ;
```

Ich werde den Parameter hps von GpiMove später erklären. GpiMove zeichnet nicht; damit wird nur die aktuelle Position eingestellt. Um die Linie zu zeichnen, werden die Felder der Struktur auf den Endpunkt eingestellt und dann GpiLine aufgerufen:

```
ptl.x = ... ;
ptl.y = ... ;

GpiLine (hps,&ptl) ;
```

Die Funktion GpiLine bewegt auch die aktuelle Position auf den angegebenen Punkt.

Es ist eindeutig ziemlich umständlich, vor dem Aufruf von GpiMove oder GpiLine die beiden Felder der Struktur POINTL zu belegen. Sie werden jedoch feststellen, daß die Verwendung einer Struktur für die Darstellung von Punkten für die Syntax der GPI-Funktionen eine schöne saubere Konsistenz bedeutet. So hat beispielsweise die Funktion GpiQueryCurrentPosition, die die aktuelle Position abfragt, die gleiche Syntax wie GpiMove:

```
GpiQueryCurrentPosition (hps,&ptl) ;
```

Bei der Rückkehr von der Funktion enthält die Struktur ptl die Koordinaten der aktuellen Position.

Die Verwendung eine Struktur für die Punkte ermöglicht eine Syntaxkonsistenz auch bei den GPI-Funktionen, die mehrere Punkte erfordern. Eine solche Funktion ist GpiPolyLine, die eine Reihe von Linien ausgehend von der aktuellen Position zeichnet. Ein Array von POINTL-Strukturen wird so definiert:

```
POINTL aptl [15] ;
```

Für den Aufruf von GpiPolyLine werden die Anzahl der Punkte und der Arrayname benötigt:

```
GpiPolyLine (hps,15L, aptl) ;
```

Dieser Aufruf von GpiPolyLine ist im wesentlichen identisch mit:

```
for (i = 0 ; i < 15 ; i++)
GpiLine (hps,aptl + i) ;
```

Die Punkte bei GPI-Funktionen werden in »Weltkoordinaten« angegeben. Obwohl GPI mehrere Umsetzungsmöglichkeiten von Welt- in Pixelkoordinaten bietet, werde ich hier noch nicht auf diese Umsetzungen eingehen. Standardmäßig sind die Weltkoordinaten Einheiten von Pixeln relativ zur linken unteren Ecke des Fensters. Werte auf der X-Achse (horizontal) erhöhen sich nach rechts; Werte auf der Y-Achse (vertikal) erhöhen sich nach oben.

CACHEDPS.C verwendet die Funktion GpiCharStringAt um einen Textstring anzuzeigen:

```
GpiCharStringAt (hps, &ptl,lTextLength,cText) ;
```

Unter normalen Umständen wird die Grundlinie der linken Seite des ersten Zeichens an dem Punkt angesetzt, der in der Struktur ptl angeben wird. Der Parameter cText ist ein Zeiger auf ein Zeichenarray, und lTextLength ist die Länge des Strings. Der folgende Code zeigt zum Beispiel den Text »Hello« am Punkt (10, 10) an:

```
ptl.x = 10 ;
ptl.y = 10 ;

GpiCharStringAt (hps, &ptl,5L,«Hello«) ;
```

Standardmäßig verwendet GPI als Schrift die normale Systemschrift, die gleiche Schrift also, die in den Titelzeilen, Menüs und Dialogboxen des Presentation Managers verwendet wird.

GPI-Attribute

CACHEDPS.C ruft zwei Funktionen auf, um die Attribute einzustellen, die GPI beim Zeichnen verwendet. Noch bevor irgendetwas gezeichnet wird, ruft das Programm GpiSetColor auf:

```
GpiSetColor (hps,CLR_RED) ;
```

Das bewirkt, daß jeder Text und alle Zeilen, die nach dieser Funktion aufgerufen werden, die Farbe rot erhalten. Bevor die vier Zeilen angezeigt werden, ruft CACHEDPS.C eine weitere Attributfunktion auf, um den Linientyp gepunktet (dot) einzustellen:

```
GpiSetLineType (hps, LINETYPE_DOT) ;
```

GPI verfügt auch über Funktionen, die Informationen vom System abfragen. Eine der Abfragefunktionen, die in CACHEDPS.C verwendet wird, ist GpiQueryTextBox. Diese Funktion füllt ein POINTL-Array mit den Positionskoordinaten der vier Ecken eines imaginären Rahmens, der diesen speziellen Zeichenstring umgeben würde, wenn er am Punkt (0,0) ausgegeben worden wäre. Das Programm CACHEDPS verwendet diese Informationen, um vier Linien zu zeichnen.

Zu Beginn der Verarbeitung von WM_PAINT ruft CACHEDPS.C die Funktion GpiCreateLogColorTable auf. In der Version des Presentation Managers, die ich für

```
hps = WinBeginPaint (hwnd, NULL,NULL) ;
    .
    .
    .
< hier werden GPI-Funktionen verwendet >
    .
    .
    .
WinEndPaint (hps) ;
```

Ein Cached-Mikro-PS hat mehrere wichtige Charakteristiken. Erstens ist ein Cached-Mikro-PS immer mit dem Bildschirm verbunden, oder genauer mit einem Fenster auf dem Bildschirm. Dadurch kann der Cached-Mikro-PS einfach verwendet werden (WinGetPS benötigt nur einen Parameter), er ist aber auch auf den Bildschirm beschränkt.

Zweitens werden alle Attribute des Präsentationsbereichs auf ihre Standardwerte eingestellt, wenn mit WinGetPS oder WinBeginPaint eine Handle angefordert wird. Darum stellt CACHEDPS.C die Farbe auf rot und den Linientyp auf gepunktet ein, wenn es die Meldung WM_PAINT verarbeitet.

Schließlich zeichnet ein Programm, daß einen Cached-Mikro-PS verwendet, gewöhnlich in Pixeleinheiten. Das kann durch einige Aufrufe bestimmter GPI-Funktionen geändert werden, es ist jedoch nicht so einfach, als wenn ein Programm mit GpiCreatePS seinen eigenen Präsentationsbereich anlegt.

Windows-Programmierer werden feststellen, daß CACHEDPS ähnlich strukturiert ist, wie ein Windows-Programm. Wenn Sie ein Programm von Windows auf den Presentation Manager umsetzen, werden Sie wahrscheinlich den Cached-Mikro-PS verwenden, um die Umsetzung zu erleichtern.

Die Meldung WM_PAINT

CACHEDPS erledigt alle Zeichenvorgänge bei der Verarbeitung der Meldung WM_PAINT. Die korrekte Handhabung der Meldung WM_PAINT ist von enormer struktureller Bedeutung, sowohl in Windows- als auch in Presentation Manager-Programmen.

Eine Fensterprozedur erhält die Meldung WM_PAINT immer dann, wenn ein Fensterbereich ungültig geworden ist, was bedeutet, daß dieser Bereich nicht länger das enthält, was das Programm ursprünglich dort gezeichnet hat. Wenn ein Teil eines Fenster bedeckt worden ist und dann wieder frei wird, schickt der Presentation Manager die Meldung WM_PAINT an das Fenster. Eine Fensterprozedur kann die Meldung WM_PAINT auch erhalten, wenn die Fenstergröße verändert wurde.

Auch wenn ein Presentation Manager-Programm fast jederzeit in einem Fenster zeichnen kann, muß das Programm bei Erhalt der Meldung WM_PAINT jedoch auch fähig sein, das ganze Fenster zu aktualisieren. Viele Programmierer bewerkstelligen dies, indem sie alle Fensterausgaben bei Erhalt der Meldung WM_PAINT durchführen und sonst nicht. Wie Sie noch sehen werden, ist die Behandlung der Meldung WM_PAINT bei einem normalen PS einfacher.

Der Mikro-PS

Da der Cached-Mikro-PS immer mit einem Fenster auf dem Bildschirm verbunden ist, kann ein Cached-Mikro-PS nicht dazu verwendet werden, auf einem anderen Ausgabegerät zu zeichnen, zum Beispiel einem Drucker oder Plotter. Um einen Drucker oder Plotter zu verwenden, müssen Sie entweder einen Mikro-PS oder einen normalen PS anlegen. Ein Programm kann einen Mikro-PS auch zum Zeichnen im Programmfenster verwenden. Das Programm MICROPS in *Listing 2* zeigt, wie dies gemacht wird.

MICROPS.C legt den Präsentationsbereich an, indem es GpiCreatePS bei der Bearbeitung der Meldung WM_CREATE in ClientWndProc verwendet. Die Angabe GPIT_MICRO in der Funktion GpiCreatePS bedeutet für GPI, daß ein Mikro-PS angelegt werden soll. GpiCreatePS benötigt auch noch eine Handle für einen Displaykontext, mit dem der Präsentationsbereich verbunden wird. Bei einem Bildschirmfenster wird diese Handle für den Displaykontext mit WinOpenWindowDC abgefragt:

```
hdc = WinOpenWindowDC (hwnd) ;
```

Der Präsentationsbereich beinhaltet eine imaginäre Zeichenoberfläche, die Präsentationsseite genannt wird (presentation page). (Ich verspreche, daß das der letzte neue Ausdruck ist, den ich hier verwende.) Die Größe und Einheiten der Präsentationsseite werden mit der Funktion GpiCreatePS definiert. Die Größe der Seite wird mit der Struktur SIZEL eingestellt. Wenn die Felder cx und cy auf 0 gesetzt werden, erhält die Seite die gleiche Größe, wie der ganze Bildschirm. Der Name PU_PELS gibt an, daß die Koordinaten der Seite in Pixeleinheiten angegeben werden. Stattdessen kann auch einer der Namen in *Tabelle 1* für andere Einheiten verwendet werden.

Name für Seiteneinheiten	Einheiten
PU_ARBITRARY	Pixel (mit Anpassung)
PU_PELS	Pixel
PU_LOMETRIC	0.1 Millimeter
PU_HIMETRIC	0.01 Millimeters
PU_LOENGLISH	0.01 Inch
PU_HIENGLISH	0.001 Inch
PU_TWIPS	1/1440 Inch

Tabelle 1: *Die Namen, die in der Funktion* GpiCreatePS *zur Einstellung der Einheiten der Präsentationsseite verwendet werden können.*

diesen Artikel verwendet habe, ist die Standardhintergrundfarbe schwarz und die Standardvordergrundfarbe weiß. Die Funktion GpiErase löscht das Fenster also auf schwarz. Der Aufruf von GpiCreateLogColorTable in CACHEDPS kehrt dies um, so daß die Standardhintergrundfarbe weiß und die Standardvordergrundfarbe schwarz ist. Ob diese Funktion auch noch in der endgültigen Version des Presentation Managers benötigt wird, ist momentan noch nicht sicher.

Die Aufrufe von GpiRestorePS, GpiSavePS und GpiResetPS in CACHEDPS.C bei der Meldung WM_PAINT sind Patches, die einige Fehler in meiner Version des Presentation Managers vermeiden sollen. Diese Funktionen sollten nicht notwendig sein, wenn der Presentation Manager seine endgültige Form hat.

PS und DC

Der erste Parameter bei jeder GPI-Funktion in CACHEDPS.C ist die Variable hps. Diese Variable wird als Typ HPS definiert, als Handle auf einen Präsentationsbereich (presentation space oder kurz PS).

Bevor ein Presentation Manager-Programm etwas auf dem Bildschirm zeichnen kann, muß es entweder einen Präsentationsbereich anlegen oder einen Präsentationsbereich verwenden, der bereits vom Presentation Manager angelegt wurde. Wie man dies macht, hängt von der Art des Präsentationsbereichs ab, für die Sie sich entscheiden.

Ein Präsentationsbereich ist im Grunde eine interne Datenstruktur des Presentation Managers. Alle Attribute eines Präsentationsbereichs - die aktuelle Position, die dezeitigen Hinter- und Vordergrundfarben, die aktuelle Linienart, Umsetzungen und mehr - werden hier gespeichert. Bei einem normalen PS kann der Präsentationsbereich auch Sammlungen von GPI-Funktionsaufrufen aufnehmen.

Ein Windows-Programmierer wird auf den ersten Blick sicherlich annehmen, daß der Präsentationsbereich dem Windows-Displaykontext (display context: DC) entspricht, doch sie sind in Wirklichkeit gänzlich verschieden. Tatsächlich beinhaltet der Presentation Manager auch das Konzept des Displaykontexts.

Beim Presenation Manger beschreibt der Displaykontext gewöhnlich ein physisches Ausgabegerät, zum Beispiel den Bildschirm, einen Drucker oder einen Plotter. Ein Displaykontext kann auch ein Speicherbereich sein, der so verwendet wird, als ob es sich dabei um ein Ausgabegerät handelt; es kann auch eine Metadatei sein. Sie können sich einen Displaykontext am besten als zusammengesetzt aus einem Ausgabegerät und einem Gerätetreiber vorstellen.

Ein Präsentationsbereich impliziert keine Art von Ausgabegerät. Statt dessen muß ein Präsentationsbereich mit einem bestimmten Displaykontext verbunden werden. Der Grund für die Trennung von Präsentationsbereich und Displaykontext wird später deutlich werden. Die konzeptionelle Beziehung zwischen einer Anwendung, dem Präsentationsbereich und dem Displaykontext ist in *Bild 2* zu sehen.

Von den drei verschiedenen Arten von Präsentationsbereichen, die vom Presentation Manager unterstützt werden, ist der normale PS (gelegentlich auch der volle PS genannt) der vielseitigste, der Mikro-PS folgt an zweiter Stelle und der unflexibelste ist der Cached-Mikro-PS.

Um einen normalen oder Mikro-PS zu verwenden, muß ein Programm mit der Funktion GpiCreatePS einen Präsentationsbereich anlegen. Genaugenommen sind dies die beiden einzigen Arten von Präsentationsbereichen, die von GPI unterstützt werden. Der Cached-Mikro-PS ist dank der Komponente des Presentation Managers verfügbar, die die Fenster und Benutzerschnittstellenfunktionen enthält, gelegentlich auch »Win«- oder »User«-Komponenten genannt. Der Presentation Manager legt diesen Mikro-PS für Sie an.

Der Cached-Mikro-PS

Das Programm CACHEDPS in *Listing 1* verwendet einen Cached-Mikro-PS. (Nähere Informationen über Cached-Mikro-PS sind im Microsoft Operating System/2 Software Developing Kit, Presentation Manager Specification, Vol. 1, S. 210 zu finden.) Ein Programm kann eine Handle auf einen Cached-Mikro-PS erhalten, indem es WinBeginPaint oder WinGetPS aufruft. CACHEDPS.C verwendet beide Funktionen.

Die Funktion WinGetPS kann bei der Verarbeitung der Fenstermeldung verwendet werden. Wenn Sie den Präsentationsbereich nicht mehr benötigen, wird er durch einen Aufruf von WinReleasePS wieder freigegeben:

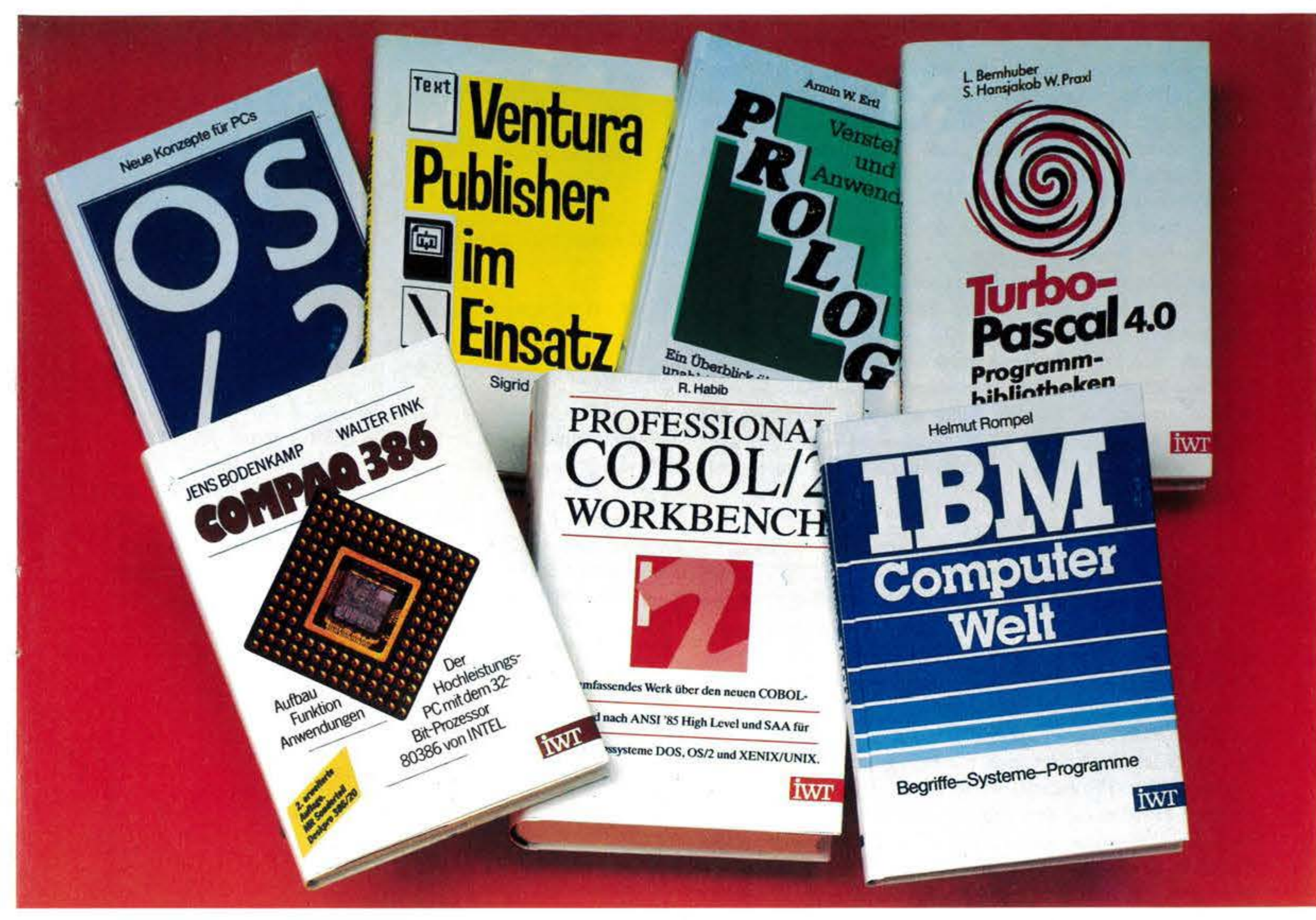
```
hps = WinGetPS (hwnd) ;
    .
    .
    .
< hier werden GPI-Funktionen verwendet >
    .
    .
    .
WinReleasePS (hps) ;
```

CACHEDPS ruft WinGetPS bei der Meldung WM_CREATE auf. Die einzige GPI-Function, die das Programm zu diesem Zeitpunkt aufruft, ist GpiQueryTestBox. CACHEDPS braucht die Dimensionen der Zeichenbox nur einmal abfragen, da die Höhe und Breite des Textstrings beim Ablauf des Programms nicht verändert werden.

Die Funktionen WinGetPS und WinReleasePS müssen bei der Bearbeitung einer Meldung paarweise aufgerufen werden. Rufen Sie niemals WinGetPS bei der Bearbeitung einer Meldung und WinReleasePS bei der Bearbeitung einer anderen auf.

Bei der Meldung WM_PAINT ruft man WinBeginPaint auf, um eine Handle für einen Cached-Mikro-PS zu erhalten, und WinEndPaint, um sie freizugeben:

Hier ist das „Update“ für Ihr Computer-Wissen.



OS/2 – ein neuer Betriebssystemstandard
Neue Konzepte für PCs

Das Buch führt in die Arbeitsweise des Betriebssystems OS/2 ein, wobei Strategien und Verfahrensweisen des neuen Betriebssystems aufgezeigt werden. Obwohl das Buch Einsteigern helfen soll, das geeignete Betriebssystem zu finden, werden zur Freude der Systemprogrammierer auch stark ins Detail gehende Fragen zu OS/2 geklärt.

1988. 200 Seiten.
Geb. DM 58,– / Fr. 58.– / S 452,–
ISBN 3-88322-205-4

Ventura Publisher im Einsatz

Dieses Buch soll dem Anwender des Programmpakets helfen, die Software professionell und wirtschaftlich einzusetzen. Gerade Funktionen, die im Bedienungshandbuch des Programms nicht sehr ausführlich dargelegt werden, sind hier detailliert beschrieben. Neun Kapitel führen systematisch von den ersten Anfängen bis zu speziellen Tips und Tricks.

1988. Ca. 400 Seiten.
Geb. Ca. DM 68,– / Fr. 68.– / S 530,–
ISBN 3-88322-199-6

Prolog – Verstehen und Anwenden
Ein Überblick über den systemunabhängigen Sprachumfang der Programmiersprache Prolog.

Ein Buch für alle, die die Einsatzmöglichkeiten einer Programmiersprache der 5. Generation besser kennenlernen oder ihre Kenntnisse in PROLOG vertiefen wollen. Auch als Nachschlagewerk für PROLOG-Anwender geeignet.

1988. 488 Seiten.
Geb. DM 68,– / Fr. 68.– / S 530,–
ISBN 3-88322-201-1

IBM-Computerwelt
Begriffe, Systeme, Programme

Das Buch bietet einen fundierten Überblick über die IBM-Computersysteme, die Datenfernübertragung (SNA-Konzept), Datenbanken und lokale Netze etc. Es zeichnet sich durch anschauliche Beschreibung der einzelnen Produkte aus.

1988. 208 Seiten.
Geb. DM 48,– / Fr. 48.– / S 374,–
ISBN 3-88322-208-9

Compaq 386
Hochleistungs-PC mit dem 32-Bit-Prozessor 80386 von INTEL. Aufbau, Funktion, Anwendungen.

Dieses Buch vermittelt einen Überblick über die Einsatzmöglichkeiten des 32-Bit-PCs 386 von COMPAQ und die Technik des 80386 Prozessors von INTEL. Dieses Gerät ist prädestiniert als Fileserver, Mehrplatzrechner oder KI-System, als Grundbaustein für eine Computergrafik- oder Desktop-publishing-station.

1988. 2., erw. Aufl. 480 Seiten.
Geb. DM 78,– / Fr. 78.– / S 608,–
ISBN 3-88322-188-0

Professional COBOL/2 Workbench
Ein umfassendes Werk über den neuen COBOL-Standard nach ANSI '85 High Level und SAA für die Betriebssysteme DOS, OS/2 und XENIX/UNIX.

Dieses Werk vermittelt eine ausführliche und gründliche Einarbeitung in die Programmiersprache COBOL. 30 Programmbeispiele und Struktogramme verhelfen dem Leser zu einem praxisorientierten Programmierstil.

1988. 896 Seiten.
Geb. DM 88,– / Fr. 88.– / S 686,–
ISBN 3-88322-200-3

Dazu Diskette lieferbar: DM 98,– */ Fr. 98,– / S 764,–
Best.-Nr. 91331101

Turbo Pascal 4.0
Programmbibliotheken und ihre Anwendung

Anhand einer Sammlung vielseitig verwendbarer Funktionen werden die Besonderheiten von Turbo Pascal 4.0 hervorgehoben. Die vorgestellten Anwendungsprogramme decken ein breites Spektrum ab: DOS-SysInfo, HDpark für 2 Platten, Filer, Plot-Ausgabe (HP-GL) am Schirm, Tiny-Lisp, usw. Tips, Tricks und andere Hinweise runden das Angebot ab.

1988. Ca. 500 Seiten.
Geb. Ca. DM 68,– / Fr. 68.– / S 530,–
ISBN 3-88322-211-9

Dazu Diskette lieferbar: DM 68,– */ Fr. 68,– / S 530,–
Best.-Nr. 91531101

*unverbindliche Preisempfehlung

Auf dem neuesten Stand zu bleiben ist das wichtigste Thema für alle, die mit Computern arbeiten. Und da gibt es eine ganze Menge Neues. Denn der Fortschritt bei Computern, Peripherie, Software und allem, was dazugehört, vollzieht sich im Laufschritt. Neue Programmiertechniken, Betriebssysteme, Software-Programme etc. kommen täglich auf den Markt. Up-to-date sein heißt deshalb, sein Wissen ständig „upzudaten".

Dabei helfen die aktuellen Fachbücher zu den aktuellen Themen. Von kompetenten Autoren aus der Praxis. Fachbücher eben, die das entsprechende Fachwissen vermitteln. Ob man nun in das Thema einsteigen, es auffrischen oder vertiefen möchte. Die Bücher dafür liefert der iwt-Verlag: Computer-Fachbücher, die weiterhelfen.

Fordern Sie den aktuellen Gesamtprospekt an.

IWT Verlag GmbH, Vaterstetten · Der Fachverlag für Information, Wissenschaft, Technologie
Wendelsteinstraße 3, 8011 Vaterstetten, Telefon (08106) 31017, Telex 5213989 iwt

AUSLIEFERUNG SCHWEIZ: THALI AG, Buchhandlung und Verlag, CH-6285 Hitzkirch, Telefon (041) 852828
AUSLIEFERUNG ÖSTERREICH: ERB-VERLAG Ges. m.b.H. + Co. KG., Amerlingstraße 1, A-1061 Wien 6, Tel. (0222) 5870526, Telex 136145

IWT

Computer-Fachbücher, die weiterhelfen.

COMMUNACTION

Die Verwendung einer Seite mit Pixelkoordinaten ist meistens die beste Vorgehensweise für einfache Grafikausgaben. Informationen, die vom Fenstersystem des Presentation Managers übergeben werden, beispielsweise die Größe des Fensters bei der Meldung WM_SIZE, wird immer in Pixeleinheiten angegeben, ganz unabhängig von der Größe der Präsentationsseite. Diese Koordinaten und Größen müssen dann mit GpiConvert in die verwendeten Seiteneinheiten umgerechnet werden.

Der Präsentationsbereich, der von GpiCreatePS übergeben wird, ist gültig, bis der Präsentationsbereich mit einem Aufruf von GpiDestroyPS gelöscht wird. MICROPS löscht den Präsentationsbereich bei der Bearbeitung der Meldung WM_DESTROY, die die letzte Meldung ist, die eine Fensterprozedur erhält. Bis dahin muß der Wert hps bei allen WM_PAINT- und WM_SIZE-Meldungen verwendet werden, deshalb wird er in einer statischen Variablen gespeichert.

Es ist zu beachten, daß MICROPS die Farbe und den Linientyp bei der Bearbeitung der Meldung WM_CREATE einstellt und nicht bei WM_PAINT. Der Mikro-PS existiert bis er explizit gelöscht wird, jedes GPI-Attribut, daß ein Programm einstellt, bleibt im Präsentationsbereich eingestellt, bis es verändert wird. Im Gegensatz dazu wird ein Cached-Mikro-PS immer auf Standardwerte zurückgesetzt, wenn ein Programm ihn anfordert.

Die Syntax der Funktion WinBeginPaint für einen Mikro-PS unterscheidet sich ein wenig von der für einen Cached-Mikro-PS. Anstatt WinBeginPaint zu verwenden, um eine Handle für den Präsentationsbereich zu erhalten, wird die vorhandene Handle als zweiter Parameter an die Funktion übergeben. (In der Version des Presentation Managers, die ich für diesen Artikel verwende, arbeiteten WinBeginPaint und WinEndPaint noch nicht richtig, deshalb stehen sie als Kommentar in MICROPS.C.)

Windows-Programmierer aufgepaßt! Sie werden es sicherlich für selbstverständlich halten, bei Verwendung eines Cached-Mikro-PS während der WM_PAINT-Meldung folgendes zu tun:

```
hps = WinBeginPaint (hwnd, NULL,NULL) ;
     .
     .
     .
< mit Standardfarben zeichnen >
     .
     .
     .
GpiSetColor (hps,CLR_RED) ;
     .
     .
     .
< mit rot zeichnen >
     .
     .
     .
WinEndPaint (hps) ;
```

Wenn Sie auf einen Mikro-PS umstellen, werden Sie den Code wahrscheinlich folgendermaßen ändern:

```
WinBeginPaint (hwnd,hps, NULL) ;
     .
     .
     .
< mit Standardfaben zeichnen >
     .
     .
     .
GpiSetColor (hps,CLR_RED) ;
     .
     .
     .
< mit rot zeichnen >
     .
     .
     .
WinEndPaint (hps) ;
```

Dies funktioniert einwandfrei bei der ersten WM_PAINT-Meldung, doch nicht bei den folgenden Meldungen, da die Farbe immer noch auf rot eingestellt ist. Beachten Sie, daß der Mikro-PS alle GPI-Attribute beibehält, bis Sie sie wieder ändern. Sie können solche Probleme vermeiden, indem Sie GpiSavePS und GpiRestorePS verwenden, um die GPI-Einstellungen zu speichern und wiederherzustellen:

```
WinBeginPaint (hwnd,hps, NULL) ;
     .
     .
     .
< mit Standardfarben zeichnen >
     .
     .
     .
GpiSavePS (hps) ;
GpiSetColor (hps,CLR_RED) ;
     .
     .
     .
< mit rot zeichnen >
     .
     .
     .
GpiRestorePS (hps,-1L) ;
WinEndPaint (hps) ;
```

Sie können die Farbe auch am Ende der WM_PAINT-Verarbeitung explizit auf den Standardwert zurücksetzen:

```
WinBeginPaint (hwnd, hps, NULL) ;
     .
     .
     .
< mit Standardfarben zeichnen >
     .
     .
     .
GpiSetColor (hps,CLR_RED) ;
     .
     .
     .
< mit rot zeichnen >
     .
     .
     .
GpiSetColor (hps, CLR_DEFAULT) ;
WinEndPaint (hps) ;
```

Beim Vergleich der Programme CACHEDPS.C und MICROPS.C zeigt sich nur eine geringfügige Strukturänderung: Zwei GPI-Attributfunktionen, die in CACHEDPS bei der Meldung WM_PAINT aufgerufen werden, werden in MICROPS bei der Meldung WM_CREATE behandelt. Eine wirkliche Strukturänderung findet erst statt, wenn Sie auf den normalen PS umsteigen.

Der normale PS

Der Mikro-PS ermöglicht einem Programm nur den Zugriff auf eine Untermenge der GPI-Funktionen. Auch wenn es eine sehr umfangreiche Untermenge ist und alle wichtigen Funktionen dazugehören, beinhaltet der Mikro-PS nicht die Funktionen, die mit GPI-Segmenten zu tun haben; diese werden nur bei einem normalen PS unterstützt.

Ein Segment (nicht zu verwechseln mit einem Speichersegment der 80286-Mikroprozessor-Architektur) ist eine gespeicherte Sammlung von Grafikzeichen- und Attributbefehlen. Man öffnet ein Segment, ruft mehrere GPI-Funktionen auf und schließt das Segment wieder. Alle Funktionen werden in dem Segment gespeichert. Man kann dann diese Befehle auf dem Ausgabegerät zeichnen lassen, indem eine von mehreren Funktionen, wie zum Beispiel GpiDrawChain, aufgerufen wird. Das Programm NORMALPS in *Listing 3* bietet Ihnen einen kleinen Vorgeschmack auf Segmente.

Die Grafikausgabe von NORMALPS hängt nur von der Größe des Fensters ab. Deshalb wird bei der Meldung WM_SIZE mit einem Aufruf von GpiOpenSegment ein Segment angelegt. Die Attribut- und Zeichenbefehle werden geanuso aufgerufen, wie in den beiden früheren Programmen. Doch diese Grafikbefehle werden nicht sofort angezeigt; sie werden nur im Segment gespeichert. Bei der Meldung WM_PAINT wird das Segment mit einem Aufruf von GpiDrawChain angezeigt. Ganz eindeutig vereinfacht dies die Bearbeitung der Meldung WM_PAINT erheblich.

Man kann in einem Programm viele Segmente anlegen; Sie werden als Teil des Präsentationsbereichs gespeichert. Ein Segment kann verkettet oder nicht verkettet sein. Die Hauptkette besteht aus allen verketteten Segmenten; GpiDrawChain zeichent alle Segmente in der Hauptkette.

Nicht verkettete Segmente können von anderen Segmenten aufgerufen werden. Wenn ein nicht verkettetes Segment aufgerufen wird, können die Positionskoordinaten, die in allen Zeichenkommandos verwendet werden, umgesetzt, skaliert oder rotiert werden, was besonders für plastische Grafiken sehr nützlich ist.

Man kann Segmente bearbeiten, die Reihenfolge verändern, in der die Segmente in der Hauptkette gezeichnet werden und prüfen, ob sich ein in einem Segment gezeichnetes Objekt innerhalb eines angegebenen Radius um einen bestimmten Punkt befindet. Dies ist vor allem dann sehr hilfreich, wenn man prüfen will, ob ein Objekt mit der Maus angeklickt wurde.

Neben der Unterstützung von Segmenten hat ein normaler PS einen weiteren Vorteil gegenüber dem Mikro-PS. Er ist der einzige Präsentationsbereich, der mit einem anderen Displaykontext verbunden werden kann; das bedeutet, man kann die Verbindung des Präsentationsbereichs mit dem Bildschirm lösen und ihn dann mit dem Drucker verbinden. Da die Segmente Teil des Präsentationsbereichs sind, brauchen sie nicht neu erstellt werden. Die Segmente sind sofort fertig für die Ausgabe auf dem Drucker.

Natürlich ist die Verwendung von Segmenten in NORMALPS übertrieben, wie das auch bei vielen anderen kleinen Presentation Manager-Programmen der Fall sein wird. Die Grundregel lautet, daß man den einfachsten Präsentationsbereich verwenden sollte, der für eine Aufgabe angemessen ist. Wenn Sie alles, was Sie machen wollen, mit einem Cached-Mikro-PS erreichen können, verwenden Sie ihn. Schleppen Sie nicht das komplette Reisegepäck für eine Woche auf einer Tagesreise mit sich herum.

Charles Petzold

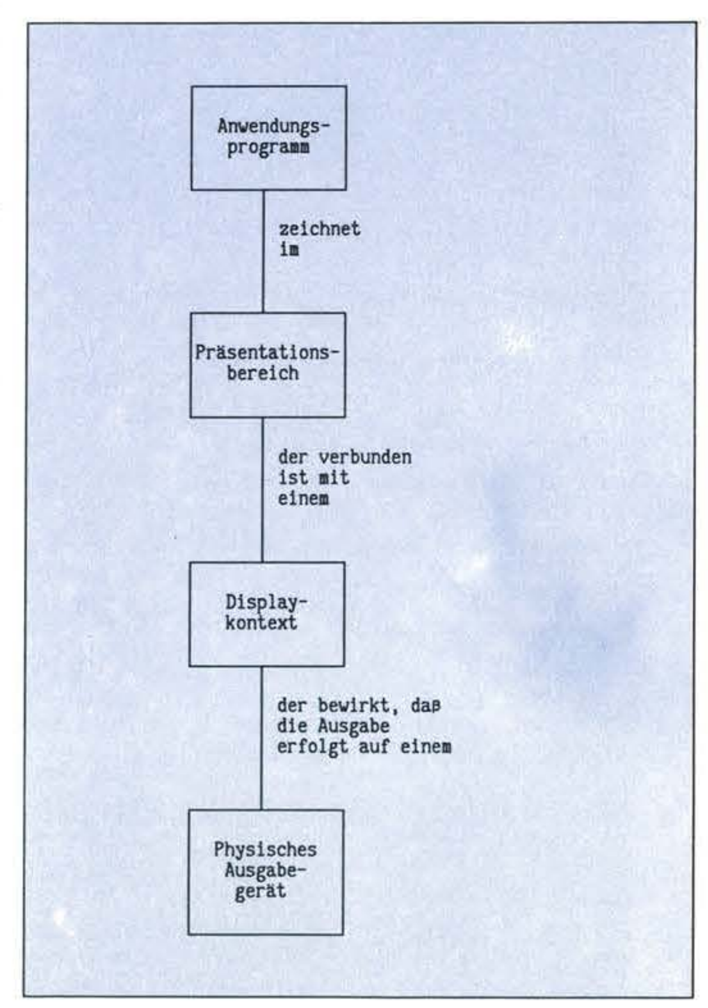


Bild 2: *Die Beziehungen zwischen einer Anwendung, dem Präsentationsbereich und dem Displaykontext.*

CACHEDPS: Die Make-Datei für CACHEDPS

```
#--------------------
# CACHEDPS Make-Datei
#--------------------

cachedps.obj : cachedps.c
     cl -c -G2sw -W3 -Zp cachedps.c

cachedps.exe : cachedps.obj cachedps.def
     link cachedps, /align:16, NUL, slibcp os2, cachedps
```

CACHEDPS.C: Der Quellcode von CACHEDPS

```
/*----------------------------------------------
   CACHEDPS.C -- Beispiel für Cached-Mikro-PS
  ----------------------------------------------*/

#define INCL_GPI

#include <os2.h>


ULONG EXPENTRY ClientWndProc (HWND, USHORT, ULONG, ULONG) ;

int main (void)
     {
     static CHAR szClientClass [] = "CachedPS" ;
     HAB    hab ;
     HMQ    hmq ;
     HWND   hwndFrame, hwndClient ;
     QMSG   qmsg ;

     hab = WinInitialize (0) ;
     hmq = WinCreateMsgQueue (hab, 0) ;

     WinRegisterClass (hab, szClientClass, ClientWndProc,
                              CS_SIZEREDRAW, 0, NULL) ;

     hwndFrame = WinCreateStdWindow (HWND_DESKTOP,
                    WS_VISIBLE | FS_SIZEBORDER | FS_TITLEBAR
                               | FS_SYSMENU    | FS_MINMAX,
                    szClientClass, "Cached Micro-PS",
                    0L, NULL, 0, &hwndClient) ;

     while (WinGetMsg (&hab, &qmsg, NULL, 0, 0))
          WinDispatchMsg (hab, &qmsg) ;

     WinDestroyWindow (hwndFrame) ;
     WinDestroyMsgQueue (hmq) ;
     WinTerminate (hab) ;

     return 0 ;
     }


ULONG EXPENTRY ClientWndProc (HWND hwnd, USHORT msg, ULONG mp1,
                                                       ULONG mp2)
     {
     static CHAR szText [] = "Graphics Programming Interface" ;
     static LONG lTextLength    = sizeof szText - 1L ;
     static LONG alColorData[] = {CLR_BACKGROUND, RGB_WHITE,
                                  CLR_NEUTRAL,    RGB_BLACK } ;
     static POINTL ptlTextStart, aptlLineStart [4],
                   aptlTextBox [TXTBOX_COUNT] ;
     static SHORT  cxClient, cyClient ;
     HPS           hps ;
     POINTL        ptl ;
     SHORT         sIndex ;

     switch (msg)
          {
          case WM_CREATE:
               hps = WinGetPS (hwnd) ;

               GpiQueryTextBox (hps, lTextLength, szText,
                                TXTBOX_COUNT, aptlTextBox) ;

               WinReleasePS (hps) ;
               break ;

          case WM_SIZE:
               cxClient = LOUSHORT (mp2) ;
               cyClient = HIUSHORT (mp2) ;

               ptlTextStart.x = (cxClient -
                       aptlTextBox [TXTBOX_BOTTOMRIGHT].x -
                       aptlTextBox [TXTBOX_BOTTOMLEFT].x) / 2 ;

               ptlTextStart.y = (cyClient -
                       aptlTextBox [TXTBOX_TOPLEFT].y -
                       aptlTextBox [TXTBOX_BOTTOMLEFT].y) / 2 ;

               for (sIndex = 0 ; sIndex < 4 ; sIndex ++)
                    {
                    aptlLineStart [sIndex] = aptlTextBox [sIndex] ;
                    aptlLineStart [sIndex].x += ptlTextStart.x ;
                    aptlLineStart [sIndex].y += ptlTextStart.y ;
                    }
               break ;

          case WM_PAINT:
               hps = WinBeginPaint (hwnd, NULL, NULL) ;

               GpiSavePS (hps) ;                        /* Patch */
               GpiResetPS (hps, GRES_ATTRS) ;           /* Patch */

               GpiCreateLogColorTable (hps, LCOL_RESET,
                    LCOLF_INDRGB, 0L, 4L, alColorData) ;

               GpiErase (hps) ;

               GpiSetColor (hps, CLR_RED) ;

               GpiCharStringAt (hps, &ptlTextStart,
                                lTextLength, szText) ;

               GpiSetLineType (hps, LINETYPE_DOT) ;

               GpiMove (hps, aptlLineStart+TXTBOX_BOTTOMLEFT) ;
               ptl.x = 0 ;
               ptl.y = 0 ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps,aptlLineStart+TXTBOX_BOTTOMRIGHT) ;
               ptl.x = cxClient ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps, aptlLineStart + TXTBOX_TOPRIGHT) ;
               ptl.y = cyClient ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps, aptlLineStart + TXTBOX_TOPLEFT) ;
               ptl.x = 0 ;
               GpiLine (hps, &ptl) ;

               GpiRestorePS (hps, -1L) ;                /* Patch */

               WinEndPaint (hps) ;
               break ;

          default:
               return WinDefWindowProc (hwnd, msg, mp1, mp2) ;
          }
     return 0L ;
     }
```

CACHEDPS.DEF: Die Moduldefinitionsdatei für CACHED-PS

```
;----------------------------------------
; CACHEDPS.DEF Moduldefinitionsdatei
;----------------------------------------

NAME          CACHEDPS
DESCRIPTION   'Beispiel für Cached-Mikro-PS (C) Ch. Petzold'
HEAPSIZE      1024
STACKSIZE     8192
EXPORTS       ClientWndProc
```

Listing 1: CHACHEDPS, CACHEDPS.C, CACHEDPS.DEF.

MICROPS: Die Make-Datei für MICROPS

```
#--------------------
# MICROPS Make-Datei
#--------------------

microps.obj : microps.c
     cl -c -G2sw -W3 -Zp microps.c

microps.exe : microps.obj microps.def
     link microps, /align:16, NUL, slibcp os2, microps
```

MICROPS.C: Der Quellcode von MICROPS

```
/*--------------------------------------
   MICROPS.C -- Beispiel für Micro-PS
  --------------------------------------*/

#define INCL_WIN
#define INCL_GPI

#include <os2.h>

ULONG EXPENTRY ClientWndProc (HWND, USHORT, ULONG, ULONG) ;

HAB  hab ;

int main (void)
     {
     static CHAR szClientClass [] = "MicroPS" ;
     HMQ    hmq ;
     HWND   hwndFrame, hwndClient ;
     QMSG   qmsg ;

     hab = WinInitialize (0) ;
     hmq = WinCreateMsgQueue (hab, 0) ;

     WinRegisterClass (hab, szClientClass, ClientWndProc,
                              CS_SIZEREDRAW, 0, NULL) ;

     hwndFrame = WinCreateStdWindow (HWND_DESKTOP,
                    WS_VISIBLE | FS_SIZEBORDER | FS_TITLEBAR
                               | FS_SYSMENU    | FS_MINMAX,
                    szClientClass, "Micro-PS",
                    0L, NULL, 0, &hwndClient) ;

     while (WinGetMsg (hab, &qmsg, NULL, 0, 0))
          WinDispatchMsg (hab, &qmsg) ;

     WinDestroyWindow (hwndFrame) ;
     WinDestroyMsgQueue (hmq) ;
     WinTerminate (hab) ;

     return 0 ;
     }

ULONG EXPENTRY ClientWndProc (HWND hwnd, USHORT msg, ULONG mp1,
                                                      ULONG mp2)
     {
     static CHAR szText [] = "Graphics Programming Interface" ;
     static HPS  hps ;
     static LONG lTextLength    = sizeof szText - 1L ;
     static LONG alColorData[] = {CLR_BACKGROUND, RGB_WHITE,
                                  CLR_NEUTRAL,    RGB_BLACK } ;
     static POINTL ptlTextStart, aptlLineStart [4],
                   aptlTextBox [TXTBOX_COUNT] ;
     static SHORT  cxClient, cyClient ;
     HDC           hdc ;
     POINTL        ptl ;
     SHORT         sIndex ;
     SIZEL         sizl ;

     switch (msg)
          {
          case WM_CREATE:
               hdc = WinOpenWindowDC (hwnd) ;

               sizl.cx = 0 ;
               sizl.cy = 0 ;

               hps = GpiCreatePS (hab, hdc, &sizl, PU_PELS |
                                   GPIF_DEFAULT | GPIT_MICRO |
                                   GPIM_NORMAL  | GPIA_ASSOC) ;

               GpiCreateLogColorTable (hps, LCOL_RESET,
                    LCOLF_INDRGB, 0L, 4L, alColorData) ;

               GpiQueryTextBox (hps, lTextLength, szText,
                                TXTBOX_COUNT, aptlTextBox) ;

               GpiSetColor (hps, CLR_RED) ;
               GpiSetLineType (hps, LINETYPE_DOT) ;
               break ;


          case WM_SIZE:
               cxClient = LOUSHORT (mp2) ;
               cyClient = HIUSHORT (mp2) ;

               ptlTextStart.x = (cxClient -
                        aptlTextBox [TXTBOX_BOTTOMRIGHT].x -
                        aptlTextBox [TXTBOX_BOTTOMLEFT].x) / 2 ;

               ptlTextStart.y = (cyClient -
                        aptlTextBox [TXTBOX_TOPLEFT].y -
                        aptlTextBox [TXTBOX_BOTTOMLEFT].y) / 2 ;

               for (sIndex = 0 ; sIndex < 4 ; sIndex ++)
                    {
               aptlLineStart [sIndex] = aptlTextBox [sIndex] ;
               aptlLineStart [sIndex].x += ptlTextStart.x ;
               aptlLineStart [sIndex].y += ptlTextStart.y ;
                    }

               break ;


          case WM_PAINT:
               /* WinBeginPaint (hwnd, hps, NULL) ; */

               GpiErase (hps) ;

               GpiCharStringAt (hps, &ptlTextStart,
                                lTextLength, szText) ;

               GpiMove (hps, aptlLineStart+TXTBOX_BOTTOMLEFT) ;
               ptl.x = 0 ;
               ptl.y = 0 ;
               GpiLine (hps, &ptl) ;
```

```
               GpiMove (hps,aptlLineStart+TXTBOX_BOTTOMRIGHT) ;
               ptl.x = cxClient ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps, aptlLineStart + TXTBOX_TOPRIGHT) ;
               ptl.y = cyClient ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps, aptlLineStart + TXTBOX_TOPLEFT) ;
               ptl.x = 0 ;
               GpiLine (hps, &ptl) ;

               /* WinEndPaint (hps) ; */

               WinValidateRect (hwnd, NULL) ;        /* Patch */
               break ;

          case WM_DESTROY:
               GpiDestroyPS (hps) ;
               break ;

          default:
               return WinDefWindowProc (hwnd, msg, mp1, mp2) ;
          }
     return 0L ;
     }
```

MICROPS.DEF: Die Moduldefinitionsdatei für MICROPS.

```
;-------------------------------------
; MICROPS.DEF module definition file
;-------------------------------------

NAME           MICROPS
DESCRIPTION    'Beispiel für Micro-PS (C) Charles Petzold'
HEAPSIZE       1024
STACKSIZE      8192
EXPORTS        ClientWndProc
```

Listing 2: MICROPS, MICROPS.C, MICROPS.DEF.

NORMALPS: Die Make-Datei für NORMALPS

```
#--------------------
# NORMALPS Make-Datei
#--------------------

normalps.obj : normalps.c
     cl -c -G2sw -W3 -Zp normalps.c

normalps.exe : normalps.obj normalps.def
     link normalps, /align:16, NUL, slibcp os2, normalps
```

NORMALPS.C: Der Quellcode von NORMALPS

```
/*--------------------------------------
   NORMALPS.C -- Beispiel für normalen PS
  --------------------------------------*/

#define INCL_WIN
#define INCL_GPI

#include <os2.h>

ULONG EXPENTRY ClientWndProc (HWND, USHORT, ULONG, ULONG) ;
HAB  hab ;

int main (void)
     {
     static CHAR szClientClass [] = "NormalPS" ;
     HMQ    hmq ;
     HWND   hwndFrame, hwndClient ;
     QMSG   qmsg ;

     hab = WinInitialize (0) ;
     hmq = WinCreateMsgQueue (hab, 0) ;

     WinRegisterClass (hab, szClientClass, ClientWndProc,
                                CS_SIZEREDRAW, 0, NULL) ;

     hwndFrame = WinCreateStdWindow (HWND_DESKTOP,
                    WS_VISIBLE | FS_SIZEBORDER | FS_TITLEBAR
                               | FS_SYSMENU    | FS_MINMAX,
                    szClientClass, "Normal-PS",
                    0L, NULL, 0, &hwndClient) ;

     while (WinGetMsg (hab, &qmsg, NULL, 0, 0))
          WinDispatchMsg (hab, &qmsg) ;

     WinDestroyWindow (hwndFrame) ;
     WinDestroyMsgQueue (hmq) ;
     WinTerminate (hab) ;

     return 0 ;
     }

ULONG EXPENTRY ClientWndProc (HWND hwnd, USHORT msg, ULONG mp1,
                                                     ULONG mp2)
     {
     static CHAR szText [] = "Graphics Programming Interface" ;
     static HPS  hps ;
     static LONG lSegmentName    = 1 ;
     static LONG lTextLength     = sizeof szText - 1L ;
     static LONG alColorData[] = {CLR_BACKGROUND, RGB_WHITE,
                                  CLR_NEUTRAL,    RGB_BLACK } ;
     static POINTL aptlTextBox [TXTBOX_COUNT] ;
     HDC            hdc ;
     POINTL         ptl, ptlTextStart, aptlLineStart [4] ;
     SHORT          cxClient, cyClient, sIndex ;
     SIZEL          sizl ;

     switch (msg)
          {
          case WM_CREATE:
               hdc = WinOpenWindowDC (hwnd) ;

               sizl.cx = 0 ;
               sizl.cy = 0 ;

               hps = GpiCreatePS (hab, hdc, &sizl, PU_PELS    |
                                        GPIF_DEFAULT | GPIT_NORMAL |
                                        GPIM_NORMAL  | GPIA_ASSOC) ;

               GpiCreateLogColorTable (hps, LCOL_RESET,
                    LCOLF_INDRGB, 0L, 4L, alColorData) ;

               GpiQueryTextBox (hps, lTextLength, szText,
                                   TXTBOX_COUNT, aptlTextBox) ;

               GpiSetDrawControl (hps, DCTL_ERASE, DCTL_ON) ;
               GpiSetDrawingMode (hps, DM_RETAIN) ;
               break ;

          case WM_SIZE:
               cxClient = LOUSHORT (mp2) ;
               cyClient = HIUSHORT (mp2) ;

               ptlTextStart.x = (cxClient -
                         aptlTextBox [TXTBOX_BOTTOMRIGHT].x -
                         aptlTextBox [TXTBOX_BOTTOMLEFT].x) / 2 ;

               ptlTextStart.y = (cyClient -
                         aptlTextBox [TXTBOX_TOPLEFT].y -
                         aptlTextBox [TXTBOX_BOTTOMLEFT].y) / 2 ;
```

```
          for (sIndex = Ø ; sIndex < 4 ; sIndex ++)
               {
               aptlLineStart [sIndex] = aptlTextBox [sIndex] ;
               aptlLineStart [sIndex].x += ptlTextStart.x ;
               aptlLineStart [sIndex].y += ptlTextStart.y ;
               }

          GpiDeleteSegment (hps, lSegmentName) ;

          GpiOpenSegment (hps, lSegmentName) ;
               {
               GpiSetColor (hps, CLR_RED) ;

               GpiCharStringAt (hps, &ptlTextStart,
                                lTextLength, szText) ;

               GpiSetLineType (hps, LINETYPE_DOT) ;

               GpiMove(hps, aptlLineStart +
                            TXTBOX_BOTTOMLEFT) ;
               ptl.x = Ø ;
               ptl.y = Ø ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps, aptlLineStart +
                             TXTBOX_BOTTOMRIGHT) ;
               ptl.x = cxClient ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps, aptlLineStart +
                             TXTBOX_TOPRIGHT) ;
               ptl.y = cyClient ;
               GpiLine (hps, &ptl) ;

               GpiMove (hps, aptlLineStart +
                             TXTBOX_TOPLEFT) ;
               ptl.x = Ø ;
               GpiLine (hps, &ptl) ;
               }
          GpiCloseSegment (hps) ;
          break ;
```

```
     case WM_PAINT:
          /* WinBeginPaint (hwnd, hps, NULL) ; */

          GpiDrawChain (hps) ;

          /* WinEndPaint (hps) ; */

          WinValidateRect (hwnd, NULL) ;          /* Patch */
          break ;

     case WM_DESTROY:
          GpiDeleteSegment (hps, lSegmentName) ;
          GpiAssociate (hps, NULL) ;
          GpiDestroyPS (hps) ;
          break ;

     default:
          return WinDefWindowProc (hwnd, msg, mp1, mp2) ;
     }
  return ØL ;
  }
```

NORMALPS.DEF: Die Moduldefinitionsdatei für NORMALPS

```
;------------------------------------------------------
; NORMALPS.DEF  Moduldefinitionsdatei
;------------------------------------------------------

NAME           NORMALPS
DESCRIPTION    'Beispiel für normalen PS (C) Charles Petzold'
HEAPSIZE       1Ø24
STACKSIZE      8192
EXPORTS        ClientWndProc
```

Listing 3: NORMALPS, NORMALPS.C, NORMALPS.DEF.

C-TOOLS

C-TOOLS Package # 1: Routinen für den Zugriff auf *sämtliche Systemeinheiten* von IBM-Personalcomputern und Kompatiblen, auf die Funktionen des *ROM-BIOS* und des Betriebssystems *DOS* für die Programmiersprache *C* im **deutsch kommentierten** *Source-Code* und im *Objekt-Code.*

Das 1. Package der C-TOOLS enthält über 100 Zugriffsroutinen auf Platte, Bildschirm, Tastatur, Drucker, Lautsprecher, den asynchronen Kommunikationsadaptor und weitere Tools. Soweit möglich, werden die Zugriffsroutinen jeweils auf *allen 3 Zugriffsebenen* zur Verfügung gestellt: auf Programmiersprachen-Ebene in C, auf der Betriebssystem-Ebene v. DOS u. auf BIOS-Ebene. Für alle BIOS-Zugriffe gibt es *assemblersprachliche Schnittstellen*, die auch mit anderen Programmiersprachen verwendet werden können.

Außerdem werden Ihnen unterschiedliche Verfahrenstechniken erklärt, z.B. für *schnelle, störungsfreie Bildschirmausgaben, Bildschirmfenster, Scrolling* etc.; Sie erhalten ein *Synthesizer-Programm*, mit dem Sie auf Ihrer Tastatur beliebige *Tonmuster* oder *Melodien* spielen und diese dann direkt in Ihr Programm einbauen können; Sie erhalten *Druckroutinen* für millimetergenaues Drucken in Vordrucke und für Graphik-Drucken u.v.m.

Preis: 632,70 DM

Die C-Tools sind nicht nur *direkt einsetzbar* (für **die meisten C-Compiler** z.B. Lattice-C und Microsoft-C), sondern verstehen sich auch als Know how-Tools:

Ausführliche *Begleitdokumentationen* liefern Ihnen detaillierte Informationen und erklären jedes Statement der C-TOOLS.

C-TOOLS Package # 2:

Datenorganisation und Speicherkonzepte, Sortierverfahren, Suchverfahren, Filter für die Programmiersprache C im deutsch kommentierten Source- und Objektcode.

Das Package # 2 enthält in der vorliegenden Version Routinen für:

- interne Datenorganisation/Speicherkonzepte: Listen, Stacks, Hashing inclusive aller Grundoperationen (z.B. Element einfügen, löschen, Position ermitteln etc.).
- Dateiorganisation und -zugriffe: sequentielle, "hashed" und indizierte Dateien
- Arbeitsspeicher- interne, externe und intern/extern-kombinierte Sortierverfahren
- Filter (z.B. variable lexikalische Sortierung, Dupletten-Filter, Dateiverschlüsselung, Spaltenanordnung etc.) **Preis: 855.– DM**

C-TOOLS Package # 3: Ein Generator für dialogorientierte Programmsysteme incl. **Windowing**.

Die überwiegende Anzahl von Anwendersystemen ist heute dialogorientiert. Die Gestaltung der Benutzerschnittstelle ist in erster Linie verantwortlich für die sog. Benutzerfreundlichkeit eines Programmsystems und bestimmt damit maßgeblich dessen Marktchancen. Die Gestaltung d. Benutzerschnittstellen wird deshalb immer trickreicher u. komfortabler. Die Programmierung solcher dialogorientierter Programmsysteme verlangt jedoch dem Programmierer einen großen Aufwand an Zeit u. Arbeit ab. Diese Programmierarbeiten erheblich zu reduzieren, ist die Aufgabe eines Dialogsystem-Generators.

Der Dialogsystem-Generator kann: Graphik- u. Textmodus, Manipulation der Bildschirmattribute, Windows/Pull-Down-Menues, Ein-/Ausgabefelder, Cursormanagement, Dialog- u. Aktionssteuerung.

Preis: 855.– DM

Unterstützte Graphik-Karten: CGA, Hercules, EGA, Olivetti, IBM Professional u.a.

C-Tools Package # 4: Graphik

Das Package enthält die wesentlichen grafischen Grundfunktionen in Quell- u. Objektcode zusammen m. ausführlichen Kommentaren innerhalb u. außerhalb d. Listings. Über 100 Einzelfunktionen in C u. Assembler für

- die Initialisierung der Grafikkarten;
- schnelles Zeichnen von Geraden, Kreisen, Ellipsen, Kreis- und Ellipsenbögen;
- das Ausfüllen von Polygonen (Fill) und konvexen Figuren (Paint) mit unterschiedlichen Füllmustern;
- die Definition von Windows und Viewports;
- die Erzeugung v. Kurven m. Hilfe kubischer B-Splines;
- Textausgaben im Grafikmodus.

z.Z. werden die folgenden Karten unterstützt: Hercules, Olivetti monochrom, CGA, EGA.

Preis: 855,– DM

C-Trainer

Lernen Sie C richtig von Anfang an!

Besonders geeignet für Schulungszwecke und C-Anfänger (C-Interpreter mit Tutorial-Programmierbuch – ein kompletter C-Lernkurs). Der C-Trainer ist eine neue, sehr effektive Methode, um C zu lernen oder sein Wissen zu erweitern. Der C-Trainer besteht aus drei Teilen: Turorial-Buch, C-Interpreter und eine C-Programmbibliothek.

Der C-Interpreter erlaubt eine hervorragende Kontrolle über die Ausführung eines C-Programmes und besitzt große Vorteile bei der Entwicklung von C-Programmen. Das Programm kann an einem beliebigen Punkt gestoppt werden, die Werte aktiver Variablen können eingesehen und geändert werden.

Programmänderungen sind sofort und ohne Compilieren und Linken möglich.

Separater oder gemeinsamer Trace für Funktionsaufrufe, Statements und Expressions ist möglich.

Verfügbar für: IBM/PC 333,– DM, Macintosh 333,– DM, Sun 561,– DM, MicroVAX (Unix o. VMS) 561,– DM, Pyramid 1473,– DM, VAX 11/700 (Unix o. VMS) 1008,– DM,

(Alle Preise zzgl. Verpackung und Versand).

Der C-Trainer ist ein Produkt der Catalytix Corp.

Die C-Tools sind Produkte des:

ECO Institut, Landshuter Straße 37, D-8400 Regensburg, Telefon (09 41) 70 04 25-26

Die Möglichkeiten der OS/2-Semaphore:

Koordination von Threads mit Semaphoren

OS/2-Semaphore sind ein mächtiger Mechanismus, um mehrere gleichzeitig ausgeführte Threads zu koordinieren. Ein häufiger Anwendungsfall ist die Zugriffssteuerung auf Programmteile und auf Ressourcen, die von diesen Teilen benötigt werden. Ein weiterer Anwendungsfall für Semaphore ist das Signalisieren eines Threads an einen anderen, wenn ein Ereignis stattgefunden hat wie zum Beispiel die Beendigung einer Ein-/Ausgabeoperation. OS/2-unterstützt beide Semaphor-Anwendungsfälle.

Es gibt vier Arten von Semaphoren in OS/2: *exklusive, nichtexklusive, RAM-* und *FSRam-*(Fast-Save-RAM)-Semaphore. FSRam-Semaphore sind neu seit der Version 1.1 von OS/2. Nichtexklusive System- und RAM-Semaphore können für die Zugriffssteuerung und das Signalisieren verwendet werden; exklusive und FSRam-Semaphore sind nur für die Zugriffssteuerung vorgesehen. FSRam-Semaphore sind mittelmäßig sicher, und ihre Geschwindigkeit ist vergleichbar mit der von RAM-Semaphoren.

Systemsemaphore können von Threads verschiedener Prozesse, die keinen gemeinsamen Speicher besitzen, verwendet werden und sie bieten gegenüber den anderen beiden einige Sicherheitsvorteile. Zukünftige Versionen von OS/2 werden weitere Schutzmechanismen und die Netzwerkunterstützung für Systemsemaphore bieten.

Wenn der Zugriff auf wiederverwendbare Ressourcen wie eine Datei, eine Datenstruktur oder des Bildschirms geregelt werden muß, so ordnet man der Ressource ein Semaphor zu und erlaubt immer nur einen Semaphor-Besitzer. Ein Thread erhält die Kontrolle über das Semaphor durch den Funktionsaufruf von `DosSemRequeset` oder `DosFSRamRequest`. Ist das Semaphor frei, erhält der Aufrufer den Rückgabewert 0 und der Thread kontrolliert das Semaphor. Ist das Semaphor schon belegt, kehrt der Aufruf entweder sofort zurück (wenn der Timeout-Parameter 0 ist) bzw. wartet n Millisekunden (Timeout = n) oder wartet undefiniert (Timeout = 1). Man sagt, der Thread ist blockiert, wenn er auf den Abschluß einer Semaphoranfrage wartet. Wenn der aufrufende Thread die Kontrolle über ein Semaphor erhält, bevor die Timeout-Periode vorüber ist, erhält er sofort einen Rückgabewert von 0. Andernfalls wartet er, bis die Timeout-Periode vorüber ist, und erhält einen Wert ungleich 0. Ein Thread gibt die Kontrolle über ein Semaphor mit den Aufrufen `DosSemClear` oder `DosFSRamSemClear` ab.

FSRam- und exklusive Systemsemaphore (Systemsemaphore, bei denen die `NoExclusive`-Option beim Erzeugen des Semaphors mit `DosCreateSem` nicht angegeben wurde) eignen sich gut für die Zugriffststeuerung. RAM-Semaphore und nichtexklusive Systemsemaphore erlauben auch die Zugriffssteuerung, Sie sollten aber bei der Verwendung einige Regeln befolgen.

Semaphortyp	Verwendung		Schutz	Leistung
Exklusives System-Semaphor	Exklusiver Zugriff	Zwischen Threads in verschiedenen Prozessen	Einige Sicherheit wird von OS/2 gewährleistet	Mäßig
Nicht-exclusives System-Semaphor	Exklusiver Zugriff	Zwischen Threads in verschiedenen Prozessen		
	Signalisieren			
RAM	Exklusiver Zugriff	Zwischen Threads eines Prozesses	Minimaler Schutz, nicht vom Betriebs-system verwaltet	Sehr groß
	Signalisieren	Zwischen Threads im selben oder einem anderen Prozeß		
FSRam (Fast-Safe Ram)	Exklusiver Zugriff	Zwischen Threads in verschiedenen Prozessen	Etwas Schutz, teilweise von OS/2 verwaltet	Groß

Tabelle 1: *Übersicht der Semaphortypen.*

Ganz gleich, ob Semaphore für die Zugriffssteuerung oder das Signalisieren verwendet werden, sie können die Unannehmlichkeiten eines *Deadlocks*, der Verletzung der Zugriffssteuerung, oder eines Prozeßabbruchs hervorrufen, während sie die Kontrolle über Ressourcen besitzen. Durch die sorgfältige Analyse der Anwendung, der Kenntnisse der OS/2-Semaphor-Funktionen und durch Befolgen einiger Regeln können solche Probleme verhindert werden.

Semaphor-Typen

Bevor Sie ein Programm mit Semaphoren schreiben, sollten Sie sich zwei Fragen beantworten. Dient das Semaphor der Zugriffssteuerung oder dem Signalisieren? Ist es für verschiedene Threads des gleichen oder verschiedener Prozesse bestimmt? Die Antwort auf diese beiden Fragen klären, welche Art von Semaphor Sie benützen müssen, und mit welchen Systemfunktionen sie angesprochen werden. Für die Zugriffssteuerung und die Synchronisation verschiedener Threads eines Prozesses sollten Sie RAM-Semaphore verwenden, für die Zugriffssteuerung von Threads verschiedener Prozesse FSRam-Semaphore und für das Signalisieren zwischen Threads verschiedener Prozesse die nichtexklusiven Systemsemaphore oder RAM-Semaphore.

Alle OS/2-Semaphor-Operationen basieren auf einer Semaphor-Handle, die 32 Bit groß ist. Im Falle des RAM- und des FSRam-Semaphors ist die Handle die Adresse der Datenstruktur. Im Falle des Systemsemaphors ist es der Wert, der von den Funktionsaufrufen `DosCreateSem` und `DosOpenSem` zurückgegeben wird.

```
if( rc=DosCreateSem( NoExclusive, &sysSem, &semName))
    if( rc == ERROR_ALREADY_EXISTS)
        DosOpenSem( &sysSem, &semName) ;
    else
            <...Fehler...>

<...Verwendung des Semaphors...>

    DosCloseSem( sysSem) ;
```

Listing 1: *Die Verwendung von Systemsemaphoren.*

RAM-Semaphore

RAM-Semaphore dienen hauptsächlich zur Koordination der Threads eines Prozesses. Sie sind schnell, da sie die einfache Datenstruktur eines Doppelworts haben, und wenig Verwaltung oder Schutz durch OS/2 erhalten.

RAM-Semaphore können zwischen verschiedenen Prozessen verwendet werden, die gemeinsamen Speicher benutzen, der durch `DosAllocShrSeg` angefordert wurde. RAM-Semaphore können aber nicht für die Zugriffssteuerung zwischen Prozessen verwendet werden, da OS/2 ein RAM-Semaphor nicht freigibt, wenn der Besitzer, der es kontrolliert hat, endet. Sie eignen sich gut für die Zugriffssteuerung zwischen Threads desselben Prozesses und können zum Signalisieren innerhalb oder zwischen Prozessen dienen.

Ein RAM-Semaphor ist ein Doppelwort-Speicherplatz, dessen Inhalt mit 0 (zurückgesetzt/nicht kontrolliert) initialisiert sein muß, bevor es als Semaphor verwendet wird. Es sollten anschließend folgende Funktionen benutzt werden: `DosSemClear`, `DosSemRequest`, `DosSemSet`, `DosSemSetWait`, `DosSemWait` oder `DosMuxSemWait`.

Systemsemaphore

Systemsemaphore sind die flexibelsten, sichersten und am leichtesten zu benutzenden Semaphore, aber ein Problem bei der Benutzung ist der Geschwindigkeitsnachteil. Sie brauchen keinen gemeinsamen Speicher, und ihr Besitz endet, wenn ihr Besitzer beendet wird. Deshalb eignen sie sich gut für die Zugriffssteuerung zwischen Prozessen.

Jeder Prozeß, der ein Systemsemaphor benötigt, muß von OS/2 mit den Funktionen `DosCreateSem` oder `DosOpenSem` eine Handle anfordern. Der Prozeß liefert einen mit 0 abgeschlossenen Semaphornamen, im gleichen Format wie einen Dateinamen einer Datei im Unterverzeichnis `\SEM\`, zum Beispiel `\SEM\RESOURCE.LCK`. OS/2 gibt dann die Handle für den weiteren Zugriff zurück. Der Prozeß gibt auch an, ob nichtbesitzende Prozesse den Zustand des Semaphors ändern können. Systemsemaphore werden in der Regel zum Signalisieren nicht exklusiv und für die Zugriffssteuerung exklusiv erzeugt.

```
long resourceSem = 0;

thread1()
{
    .
    .
    .
    DosSemRequest( &resourceSem, -1L);

    <... lesen/ändern der Ressource ...>

    DosSemClear( &resourceSem);
    .
    .
    .
}

thread2()
{
    .
    .
    .
    DosSemRequest( &resourceSem, -1L);

    <... schreiben/ändern der Ressource...>

    DosSemClear( &resourceSem);
    .
    .
    .
}
```

Listing 2: *Zwei Threads haben nacheinander exklusiven Zugriff auf eine Ressource.*

`DosCreateSem` wird verwendet, wenn das Semaphor noch nicht existiert (existiert es bereits, erhält der Aufrufer einen Fehlercode), und das Semaphor wird als nicht kontrolliert initialisiert. `DosOpenSem` wird verwendet, wenn ein anderer Prozeß das Semaphor erzeugt hat; falls es nicht existiert, wird ein Fehlercode zurückgegeben. Wenn in Ihrer Anwendung ein Prozeß ganz bestimmt zuerst läuft, kann er `DosCreateSem` verwenden. Spätere Prozesse können dann `DosOpenSem` aufrufen. Ist die Reihenfolge der Prozesse unbekannt, kann jeder `DosCreateSem` aufrufen, und falls der Aufruf einen Fehler zurückmeldet, weil das Semaphor bereits existiert, `DosOpenSem`.

Wenn ein Prozeß ein Systemsemaphor nicht mehr benötigt, sollte er `DosCloseSem` aufrufen. Das Systemsemaphor wird gelöscht, wenn alle Prozesse, die das Semaphor benutzen, `DosCloseSem` aufgerufen haben. Wird ein Prozeß mit offenem Systemsemaphor beendet, dann schließt das System dieses (*Listing 1*).

Ein Thread sollte normalerweise kein Semaphor anfordern, das er bereits kontrolliert. Dies würde ihn blockieren, bis er selbst das Semaphor freigibt, was einem Deadlock entspricht.

Ein Systemsemaphor, das mit der exklusiven Option erzeugt wurde, blockiert einen Thread nicht, wenn er einen `DosSemRequest`-Funktionsaufruf durchführt, während er das Semaphor kontrolliert. Statt dessen wird der Nutzungszähler des Semaphors erhöht. Bei jedem Aufruf der Funk-

tion DosSemClear wird der Nutzungszähler vermindert, und wenn er 0 ist, wird das Semaphor freigegeben. Beachten Sie, daß ein solches Semaphor nicht einem klassischen Semaphor entspricht, das später in diesem Artikel noch beschrieben wird. Will ein anderer Thread ein exklusiv vergebenes Semaphor mit DosSemRequest anspricht, wird er blockiert und muß warten bis der Nutzungszähler 0 ist.

Nichtexklusive Systemsemaphore verhalten sich wie RAM-Semaphore, wenn Sie mit DosSemClear, DosSemRequest, DosSemSet, DosSemWait oder DosMuxSemWait angesprochen werden.

FSRam-Semaphore

FSRam-Semaphore erlauben die Zugriffsteuerung zwischen Prozessen und bieten fast die Geschwindigkeit von RAM-Semaphoren in Kombination mit der Sicherheit von Systemsemaphoren. Sie benutzen eine einfache Datenstruktur und benötigen geringen Systemaufwand. Ihre Ausführungsgeschwindigkeit ist deshalb sehr gut. Da sie Daten zur Verwaltung der Semaphorbesitzer führen, erlauben FSRam-Semaphore der Funktion DosExitList das Freigeben einer kontrollierten Ressource, wenn der Prozeß endet.

Ebenso wie die exklusiven Systemsemaphore erlauben FSRam-Semaphore rekursive DosFSRamSemRequest-Zugriffe durch das Hochzählen eines Nutzungszählers, und sie unterstützen DosFSRamSemClear, indem sie den Nutzungszähler vermindern. Wird der Nutzungszähler 0, ist das Semaphor frei. (Beachten Sie aber bitte wieder, daß ein Semaphor sich von einem klassischen Zählsemaphor unterscheidet.) DosFSRamSemRequest und DosFSRamSemClear sind die einzigen in OS/2 verfügbaren Funktionen für FSRam-Semaphore.

FSRam-Semaphore befinden sich im gemeinsamen Speicher der kooperierenden Prozesse und haben ein Längenfeld von 12; alle anderen Felder sind vor dem Gebrauch auf 0 initialisiert.

Zugriffssteuerung

In OS/2 können Semaphore gewährleisten, daß nur ein Thread zu einer Zeit Zugriff zu einer geschützten Ressource hat. Dies wird durch die Zuordnung eines Semaphors zu der geschützten Ressource erreicht und mit den Programmierkonventionen erreicht, daß jedes Programmteil, das diese geschützte Ressource benötigt, vorher einen DosSemRequest-Funktionsaufruf (oder DosFSRamSemRequest) und nachher einen DosSemClear-Funktionsaufruf (oder DosFSRamSemClear) durchführen muß. Es ist wichtig, daß jeder beteiligte Thread sich an dieses Schema hält, ansonsten treten zeitabhängige Fehler auf.

Der Thread, dessen Semaphoranforderung ohne Fehlermeldung erledigt wird, kontrolliert das Semaphor und alle anderen Threads, die einen ähnlichen Aufruf durchführen, sind blockiert. Dies ermöglicht dem kontrollierenden Thread den exklusiven Zugriff zu dem Programmteil, der die geschützte Ressource verwendet. Benötigt der Thread die geschützte Ressource nicht mehr, löscht er das Semaphor und gibt es dadurch ab. Ist ein anderer Thread wegen der Anforderung des inzwischen wieder verfügbaren Semaphors blockiert, kann er jetzt fortfahren und auf die geschützte Ressource zugreifen.

Stellen Sie sich zum Beispiel zwei Threads vor, die ein Semaphor benutzen, um die Zugriffssteuerung auf eine Ressource zu regeln. Sie verwenden vielleicht einen Programmteil wie im *Listing 2*, wo die 1 signalisiert, daß der Thread undefiniert lange wartet, bis das gewünschte Semaphor verfügbar ist.

Die vorher beschriebene Programmierkonvention sollte strikt eingehalten werden, außer wenn sie für eine spezielle Anwendung nicht benötigt wird. Sogar eine nur leichte Abwandlung kann zu einem Zugriffsfehler, Deadlock oder beidem führen.

Ein FSRam-Semaphor kann für die Zugriffssteuerung zwischen Threads verschiedener Prozesse sicher benutzt werden, wenn jeder Prozeß ein Wiederherstellen im Falle der Beendigung vorsieht, während er das Semaphor noch kontrolliert. Vor der Beendigung sollte jeder Prozeß eine ExitList-Routine ausführen, um die Integrität der geschützten Ressourcen durch den Funktionsaufruf DosExitList zu gewährleisten. Bei der Beendigung des Prozesses wird die ExitList-Routine aufgerufen. Diese sollte zuerst DosFSRamSemRequest aufrufen, um die Kontrolle über das Semaphor zu erhalten, dann die Ressource in Ordnung bringen und schließlich DosFSRamSemClear aufrufen, um die Ressourcen für die Benutzung der anderen Prozesse freizugeben.

Wird das FSRam-Semaphor von einem Thread des beendenden Prozesses kontrolliert, wenn DosFSRamSemRequest während der ExitList-Routine ausgeführt wird, wird die Thread-ID des kontrollierenden Threads zur momentanen Thread-ID und der Nutzungszähler auf 1 gesetzt. Dies erlaubt der ExitList-Routine die Ressource in einen sicheren Zustand zu versetzen und sie dann mit DosFSRamSemClear freizugeben (*Listing 3*).

RAM-Semaphore sollten nur für die Zugriffssteuerung von Threads desselben Prozesses benutzt werden. In diesem Fall sind sie effizient (da es wenig Systemoverhead gibt), leicht verwendbar (da Threads sich Speicher teilen) und sicher (da der Prozeß nur endet, wenn alle seine Threads enden).

Es ist möglich, ein nichtexklusives Systemsemaphor für die Zugriffssteuerung zu benutzen, aber alle Threads, die das Semaphor benutzen, müssen sich an die Programmierkonventionen für den aufeinanderfolgenden Zugriff halten. Nichtexklusive RAM-Semaphore verhalten sich im allgemeinen ähnlich wie RAM-Semaphore. Wenn aber der Besitzer stirbt, während er es kontrolliert, erhält der nächste Thread beim Aufruf von DosSemRequest das Sema-

```
main()
{
    .
    .
    .
    DosExitList( 1, &Cleanup);    /* Zur Exit-Liste hinzufügen*/
    .
    .
    .
}

Cleanup()
{
    if( DosFSRamSemRequest( &sem, 0L) != ERR_TIMEOUT)
    {
            <... in einen sicheren Zustand überführen ...>
            DosFSRamSemClear( &sem);
    }
    DosExitList( 3, 0); /* nächste ExitList-Routine aufrufen */
}
```

Listing 3: *Überführen einer Ressource in einen sicheren Zustand.*

phor und einen Fehlercode. Der neue Besitzer, ein richtiger Thread oder eine ExitList-Routine, kann die Ressource in Ordnung bringen und DosSemClear aufrufen.

Exklusive Systemsemaphore können zur sicheren Zugriffssteuerung zwischen verschiedenen Prozessen benutzt werden, da die Datenstruktur des Systemsemaphors eine ID des kontrollierenden Threads enthält und jeder Versuch eines Threads, den Status eines exklusiven Systemsemaphors zu ändern, die von einem ander Thread kontrolliert wird, veranlaßt eine Blockierung des Threads oder die Rückgabe eines Fehlercodes.

Wenn ein Thread im Besitz der Kontrolle eines Systemsemaphors ist und entweder er selbst endet, oder sein Prozeß beendet wird, wird in der Struktur des Semaphors ein Flag gesetzt. Der nächste Thread, der das Semaphor bei einem Funktionsaufruf von DosSemRequest erhält, bekommt den Fehlerhinweis, daß der Besitzer des Semaphors beendet ist und kann passende Wiederherstellungsmaßnahmen durchführen. Ein nachfolgender Aufruf von DosSemClear gibt das Semaphor frei und löscht das Fehlerflag.

ExitList für die Bearbeitung von Systemsemaphore, die für die Zugriffssteuerung benutzt werden, gleicht stark der ExitList-Bearbeitung von FSRam-Semaphoren.

Signale

RAM- oder nichtexklusive Systemsemaphore erlauben das Erkennen eines Ereignisses in anderen Threads. Nehmen Sie zum Beispiel den Fall, daß der Programmteil F1 in Thread 1 vor dem Programmteil F2 in Thread 2 ausgeführt werden muß und das Systemsemaphor mit der Handle sigSem mit der Option »nichtexklusiv« erzeugt wurde. Das Programm könnte aussehen wie das in *Listing 4*.

```
DosSemSet( sigSem);

thread1()
{   .
    .
    .
    F1
    DosSemClear( sigSem);
    .
    .
    .
}


thread2()
{   .
    .
    .
    DosSemWait( sigSem, -1L);
    F2
    .
    .
    .
}
```

Listing 4: *Signale zwischen Threads.*

Im ungünstigsten Fall könnte Thread 2 den Aufruf von DosSemWait ausführen, bevor Thread 1 die Ausführung von F1 beendet, wodurch er blockiert wird. Wenn der Thread 1 die Bearbeitung von F1 beendet, löscht er sigSem und der Thread 2 fährt mit der Ausführung von F2 fort.

Die Semaphorfunktionen DosSemSet, DosSemClear, DosSemWait und DosMuxSemWait dienen der Bearbeitung von Signalen. DosSemSet setzt das Semaphor, DosSemClear löscht es. DosSemWait blockiert den Thread, bis das Semaphor gelöscht ist oder bis ein optionales Timeout-Intervall verstrichen ist. DosSemSetWait ist eine untrennbare Kombination von DosSemSet gefolgt von DosSemWait. DosMuxSemWait blockiert den Thread, bis ein Semaphor aus einer Liste gelöscht oder bis ein Timeout-Intervall verstrichen ist.

Sie sollten die Signalaufrufe nicht mit exklusiven Systemsemaphoren durchführen. Nehmen Sie zum Beispiel einen Thread, der DosSemSetWait benutzt. Wird das exklusive Systemsemaphor bereits kontrolliert, wird ein Fehler zurückgegeben. Anderenfalls setzt der Thread das Semaphor und wartet, bis es gelöscht wird. Das passiert aber nicht, da kein anderer Thread es löschen kann.

Listing 5 zeigt ein Beispiel von DosSemSetWait, ähnlich dem oben erwähnten mit DosSemWait, aber diesmal in einer Endlosschleife. Der Programmteil F2 kann nur einmal für jede Bearbeitung von F1 ausgeführt werden.

Listing 6 zeigt die Verwendung von DosMuxSemWait. In diesem Fall blockiert Thread 4 bis entweder Thread 1, oder Thread 2 oder Thread 3 ihre Semaphore löschen. Thread 4 fährt fort und führt die für den signalisierenden Thread passende Aktion aus.

```
thread1()
{   .
    .
    .
    while(1)
    {
        F1
        DosSemClear( sigSem) ;
    }
    .
    .
    .
}

thread2()
{   .
    .
    .
    while(1)
    {
        DosSemSetWait( segSem, -1L) ;
        F2
    }
    .
    .
    .
}
```

Listing 5: *Die Verwendung von* `DosSemSetWait`.

Ebenso wie bei der Zugriffsteuerung ist bei der Verwendung von Semaphoren für die Signalbearbeitung eine gründliche Analyse notwendig, um Deadlocks oder Synchronisationsfehler zu vermeiden.

Ringpuffer

Stellen Sie sich vor, zwei Threads eines Prozesses kommunizieren über einen Ringpuffer. Thread 1 schreibt in den Puffer, Thread 2 liest aus dem Puffer. Der Puffer hat eine feste Länge, `bufSize`, und Zeiger auf den neuesten (`head`) und den ältesten (`tail`) Datensatz. Der Puffer ist in dem Sinne ringförmig, daß jeder Thread nach dem Zugriff auf die Position `bufSize - 1` auf Position 0 zugreift. Der Puffer ist leer, wenn `head == tail`, und er ist voll, wenn `tail == head + 1 (mod bufSize)`.

Stellen Sie sich weiter vor, daß `mutexSem`, `emptySem` und `fullSem` RAM-Semaphore sind, und daß die Befehle

```
DosSemClear(&mutexSem);
DosSemSet(&emptySem);
DosSemClear(&fullSem);
head = tail = 0;
```

ausgeführt werden, bevor die Threads starten. Dann würden die beiden Threads so aussehen wie in *Listing 7*.

Die geschützte Ressource ist in diesem Fall der Puffer, nebst seinen Variablen (`head, tail, emptySem` und `fullSem`). Jeder Thread greift auf die geschützten Ressourcen zwischen den Funktionsaufrufen `DosSemRequest` und `DosSemClear` für das ausschließlich einem vorbehaltene Semaphor `mutexSem` zu. Die Zugriffe erfolgen nacheinander und es entsteht keine Konkurrenzsituation.

```
struct {
    int numSem;
    int res1;
    unsigned long semHandle1;
    int res2;
    unsigned long semHandle2;
    int res3;
    unsigned long semHandle3;
} muxSemList ;

int muxIndex;

thread1()
{   .
    .
    .
        F1
        DosSemClear( sigSem1) ;
    .
    .
    .
}

thread2()
{   .
    .
    .
        F2
        DosSemClear( sigSem2) ;
    .
    .
    .
}

thread3()
{   .
    .
    .
        F3
        DosSemClear( sigSem3) ;
    .
    .
    .
}

thread4()
{   .
    .
    .
        muxSemList.numSem = 3 ;
        muxSemList.res1 = 0;
        muxSemList.semHandle1 = sigSem1 ;
        muxSemList.res2 = 0 ;
        muxSemList.semHandle2 = sigSem2 ;
        muxSemList.res3 = 0 ;
        muxSemList.semHandle3 = sigSem3 ;
    .
    .
    .
        DosMuxSemWait( &muxIndex, &muxSemList, -1L) ;
        switch( muxIndex) {
        case 1:
    .
    .   /* Antwort auf F1 */
    .
        case 2:
    .
    .   /* Antwort auf F2 */
    .
        case 3:
    .
    .   /* Antwort auf F3 */
    .
        }
}
```

Listing 6: *Die Verwendung von* `DosMuxSemWait`.

Schenken Sie einem netten Menschen ein MS...\etc.-Jahresabo

Das sind die Vorteile für MS.../etc-Abonnenten:

- Informationen aus erster Hand
- Fachberichte über ausgereifte Applikationen
- Umfangreiche Testberichte
- Veröffentlichung von Leserartikeln
- Neue Produkte
- Viele Fragen und Antworten zur Technik



MS..\etc · BESTELLUNG

Ja, ich bestelle die MS..\etc ab: ____________

☐ für mich/uns ☐ für umseitigen Empfänger

Für: ☐ 1 Jahr (12 Ausgaben) zum Vorteilspreis von DM 28,–

☐ 2 Jahre (24 Ausgaben) zum Vorteilspreis von DM 50,–

Postzustellung frei Haus. Vertriebskosten und Mehrwertsteuer sind im Vorteilspreis enthalten

Name (bitte in Blockschrift)

Vorname

Straße und Hausnummer

PLZ und Ort

Dieses Angebot gilt für die Bundesrep. Deutschland u. West-Berlin.
Auslandspreise:
Schweiz sfr 30,-, Österreich öS 250,- (für 1 Jahr)
Schweiz sfr 54,-, Österreich öS 450,- (für 2 Jahre)

Das Abonnement verlängert sich automatisch um ein weiteres Jahr zu den dann gültigen Preisen, wenn es nicht acht Wochen vor Ablauf gekündigt wird.

Ich bezahle mein Abonnement:

☐ Sofort nach Erhalt der Rechnung.

☐ Durch Bankeinzug.
Hiermit ermächtige ich Sie widerruflich, die von mir zu entrichtenden Zahlungen für bei Ihnen bestellte Artikel bei Fälligkeit zu Lasten meines

Kontos Nr.: ____________

BLZ.: ____________

Geldinstitut: ____________
durch Lastschrift einzuziehen. Wenn mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Geldinstituts keine Verpflichtung zur Einlösung.

Datum/Unterschrift

Vorteile für Microsoft System Journal Abonnenten mit Diskettenbestellung

- *Listings auf Diskette zum Compilieren*
- *schnelle Übernahme in Programme*
- *lieferbar auf der derzeit am weitest verbreiteten 5¼" Diskette 360 KB für IBM PC und Kompatible*

Sichern Sie sich das Kombinationspaket der gedruckten Ausgabe des Microsoft System Journal und der digitalisierten Daten auf Diskette

Microsoft System Journal Disketten – Bestellung

Ja, ich bestelle das Microsoft System Journal + Diskette ab: ________

☐ Für 1 Jahr (6 Ausgaben) zum Vorteilspreis von DM 230,–

☐ für 2 Jahre (12 Ausgaben) zum Vorteilspreis von DM 420,–.

ich bestelle nur die Diskette

☐ Einzeldiskette je Ausgabe	DM 19,80
☐ für 1 Jahr (6 Ausgaben)	DM 118,80
☐ für 2 Jahre (12 Ausgaben)	DM 220,—

Name (bitte in Blockschrift)

Vorname

Straße und Hausnummer

PLZ und Ort

Dieses Angebot gilt nur für die Bundesrepublik Deutschland und West-Berlin.

Auslandspreise: Microsoft System Journal + Diskette:

für 1 Jahr (6 Ausgaben) Schweiz sfr 230,-, Österreich öS 1 800,-
für 2 Jahre (12 Ausgaben) Schweiz sfr 420,-, Österreich öS 3 300,-.

Auslandspreise für Diskettenbestellung

Einzeldiskette	Schweiz sfr 19,80	Österreich öS 150,—
für 1 Jahr (6 Ausgaben)	Schweiz sfr 118,80	Österreich öS 950,—
für 2 Jahre (12 Ausgaben)	Schweiz sfr 220,—	Österreich öS 1 700,—

Das Abonnement verlängert sich automatisch um ein weiteres Jahr zu den dann gültigen Preisen, wenn es nicht acht Wochen vor Ablauf gekündigt wird.

Ich bezahle mein Abonnement:

☐ Sofort nach Erhalt der Rechnung.

☐ Durch Bankeinzug.

Bitte die Einzugsermächtigung auf der Rückseite ausfüllen.

Vorteile für Microsoft System Journal Abonnenten:

- *Experten-Informationen*
- *Fragen und Antworten*
- *Tips & Tricks*
- *Termine & Daten*
- *Information über Bücher & Software*
- *Tools & Routinen*
- *usw., usw., usw.*

Sichern Sie sich das Microsoft System Journal zum Abo-Vorteilspreis von DM 115,- für 1 Jahr (6 Ausgaben), oder von DM 210,- für 2 Jahre (12 Ausgaben).

Microsoft System Journal ·BESTELLUNG

Ja, ich bestelle das Microsoft System Journal ab: ________

Für:

☐ 1 Jahr (6 Ausgaben) zum Vorteilspreis von DM 115,–

☐ 2 Jahre (12 Ausgaben) zum Vorteilspreis von DM 210,–.

Postzustellung frei Haus. Vertriebskosten und Mehrwertsteuer sind im Vorteilspreis enthalten.

Name (bitte in Blockschrift)

Vorname

Straße und Hausnummer

PLZ und Ort

Dieses Angebot gilt nur für die Bundesrepublik Deutschland und West-Berlin.
Auslandspreise:
Schweiz sfr 115,- Österreich öS 850,- für 1 Jahr (6 Ausgaben).
Schweiz sfr 210,- Österreich öS 1 600,- für 2 Jahre (12 Ausgaben).

Das Abonnement verlängert sich automatisch um ein weiteres Jahr zu den dann gültigen Preisen, wenn es nicht acht Wochen vor Ablauf gekündigt wird.
Ich bezahle mein Abonnement:

☐ Sofort nach Erhalt der Rechnung.

☐ Durch Bankeinzug.
Hiermit ermächtige ich Sie widerruflich, die von mir zu entrichtenden Zahlungen für bei Ihnen bestellte Artikel bei Fälligkeit zu Lasten meines

Kontos Nr.: ____________

BLZ.: ____________

Geldinstitut: ____________
durch Lastschrift einzuziehen. Wenn mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Geldinstituts keine Verpflichtung zur Einlösung.

Datum/Unterschrift

Empfänger der Zeitschrift

Name (bitte in Blockschrift)

Vorname

Straße und Hausnummer

PLZ und Ort

Garantie

Mir ist bekannt, daß ich diese Bestellung innerhalb einer Woche bei der Bestelladresse widerrufen kann. Zur Wahrung der Frist genügt die rechtzeitige Absendung meines Widerrufsschreibens.

Ich bestätige dies durch meine zweite Unterschrift.

Datum/Unterschrift

Bitte mit 60 Pfg. freimachen

Antwort

schury praxisformulare GmbH
Bereich **Publikationen**

Postfach 270

D-8200 Rosenheim

\etc

Schenkanstoß MS...\etc.-Jahresabo

Einzugsermächtigung

Hiermit ermächtige ich Sie widerruflich, die von mir zu entrichtenden Zahlungen für bei Ihnen bestellte Artikel bei Fälligkeit zu Lasten meines

Kontos Nr.:

BLZ.:

Geldinstitut:

durch Lastschrift einzuziehen. Wenn mein Konto die erforderliche Deckung nicht aufweist, besteht seitens des kontoführenden Geldinstituts keine Verpflichtung zur Einlösung.

Datum/Unterschrift

Garantie

Mir ist bekannt, daß ich diese Bestellung innerhalb einer Woche bei der Bestelladresse widerrufen kann. Zur Wahrung der Frist genügt die rechtzeitige Absendung meines Widerrufsschreibens.

Ich bestätige dies durch meine zweite Unterschrift.

Datum/Unterschrift

Bitte mit 60 Pfg. freimachen

Antwort

schury praxisformulare GmbH
Bereich **Publikationen**

Postfach 270

D-8200 Rosenheim

Jetzt zum Abo-Vorteilspreis bestellen!

auch mit Diskette

Listings auf Diskette zum Compilieren oder zur schnellen Übernahme in Programme
Lieferbar auf $5\frac{1}{4}''$ Disketten 360 KB für IBM PC und Kompatible

Microsoft System Journal +Diskette

im Jahresabonnement (6 Ausgaben) zu DM 230,-
im 2-Jahresabonnement (12 Ausgaben) zu DM 420,-

Garantie

Mir ist bekannt, daß ich diese Bestellung innerhalb einer Woche bei der Bestelladresse widerrufen kann. Zur Wahrung der Frist genügt die rechtzeitige Absendung meines Widerrufsschreibens.

Ich bestätige dies durch meine zweite Unterschrift.

Datum/Unterschrift

Bitte mit 60 Pfg. freimachen

Antwort

schury praxisformulare GmbH
Bereich **Publikationen**

Postfach 270

D-8200 Rosenheim

Jetzt zum Abo-Vorteilspreis bestellen!



Insider-Informationen für Programmierer, Softwareentwickler, Systemdesigner und erfahrene Anwender von Microsoft-Software.

Microsoft System Journal

im Jahresabonnement (6 Ausgaben) zu DM 115,-
im 2-Jahresabonnement (12 Ausgaben) zu DM 210,-

```
thread1()
{  .
   .
   .
   <... Datensatz c holen ...>
   DosSemWait( &fullSem, -1L) ;

   DosSemRequest( &mutexSem, -1L);
   Buffer.head = c;              /* c im Puffer ablegen */
   head++;                       /* Pufferzeiger erhöhen */
   head %= bufSize;              /* Zurück zum Start */
   if((head==tail)||((tail==0)&&(head==bufSize-1)))
       DosSemSet( &fullSem);     /* setzen wenn voll */
   DosSemClear( &emptySem);      /* nicht leer */
   DosSemClear( &mutexSem);
   .
   .
   .
}
thread2()
{  .
   .
   .
   DosSemWait( &emptySem, -1L);

   DosSemRequest( &mutexSem, -1L);
   c = Buffer.tail;              /* c vom Puffer holen */
   tail++;                       /* Pufferende erhöhen */
   tail %= bufSize;              /* Zum Beginn */
   if( head==tail)
       DosSemSet( &emptySem);    /* setzen wenn leer */
   DosSemClear( &fullSem);       /* nicht voll */
   DosSemClear( &mutexSem);

   <... c benützen ...>
   .
   .
   .
}
```

Listing 7: *Beispiel für einen Ringpuffer.*

Vor dem Start der Threads ist die geschützte Ressource in einem gesicherten Zustand: `head == tail`, `emptySem` ist gesetzt und `fullSem` ist gelöscht. Jeder Thread beeinflußt den Zustand der geschützten Ressource so, daß wenn er vor der kritischen Sektion in Ordnung war, er es auch nach dem Verlassen ist. Wenn also Thread 1 einen Datensatz in den Puffer einfügt, der den letzten freien Platz belegt, dann setzt er das Semaphor `fullSem`. Thread 1 löscht immer das Semaphor `emptySem`, da er immer einen Datensatz in den Puffer einfügt. Ebenso setzt Thread 2 das Semaphor `emptySem`, wenn er den letzten Datensatz aus dem Puffer nimmt und löscht immer das Semaphor `fullSem`.

Ist der Puffer voll, setzt Thread 1 `fullSem` und blockiert bei `fullSem`. In diesem Fall wird `emptySem` nicht gesetzt und da Thread 1 außerhalb des kritischen Teiles blockiert, kann Thread 2 den kritischen Teil betreten, einen Eintrag aus dem Puffer entnehmen, `fullSem` löschen und damit dem Thread 1 die weitere Bearbeitung ermöglichen.

```
P()
{
   <... warten bis classicCountSem > 0 ...>
   classicCountSem--;
}
V()
{
   classicCountSem++;
}
```

Listing 8: *Klassisches Zählsemaphor.*

Ist der Puffer leer, setzt Thread 2 `emptySem` und blockiert bei `emptySem`. Da `fullSem` nicht gesetzt ist, und Thread 2 außerhalb des kritischen Teiles blockiert, kann Thread 1 den kritischen Teil bearbeiten, einen Eintrag in den Puffer einfügen, `emptySem` löschen und dem Thread 2 die weitere Bearbeitung ermöglichen.

Die Simulation eines klassischen Semaphors

Klassische Semaphore können leicht mit den Möglichkeiten nachgebildet werden, die die OS/2-Semaphore bieten. `ClassicCountSem` ist eine Integervariable, die abgesehen von der Initialsierung nur mit den beiden elementaren Operationen `P` und `V` angesprochen werden kann *(Listing 8)*.

Zur Simulation von `classicCountSem` und der entsprechenden Operationen `P` und `V` verwenden Sie die OS/2-Semaphore `countSem` zum Signalisieren und `mutexSem` zur Zugriffssteuerung, die beide anfänglich gelöscht sind, und die Integer-Variable `count`, die anfänglich nicht negativ ist *(Listing 9)*.

Alle Programmteile der Operationen `P` und `V`, die `count` und `countSem` ändern, liegen in sich gegenseitig ausschließenden Bereichen, die durch Aufrufe von `mutexSem` eingeschlossen werden, was das Analysieren der Routinen erleichtert.

Diese Routinen würden auch korrekt arbeiten, wenn die Referenzen auf `countSem` entfernt würden, aber `P` würde viel Zeit verbrauchen, um Bedingungen abzuprüfen, die nicht erfüllt sind. Das Warten (`DosSemWait`) auf `countSem` verhindert unnütze Aktivitäten, erlaubt aber auch `P` weiterzumachen, wenn eine `Chance` zum Beenden besteht. `P` blockiert nur auf `countSem`, wenn `count` 0 ist. Wenn `V` ausgeführt wird, wird `count` erhöht und `countSem` gelöscht, was `P` erlaubt, weiterzumachen, bis `count` erneut 0 ist. Deshalb haben alle wartenden `Ps` die Gelegenheit, die von `V` »erzeugten« Einheiten aufzubrauchen.

Verwaltung

Das Zeitverhalten und die Synchronisierung der Threads mit Semaphoren wird durch die Aktivitäten der anderen Threads im System, zum Beispiel Priorität und Position in

der Liste der blockierten Threads, beeinflußt. Wenn mehrere Threads auf ein Semaphor warten, sollte Ihre Anwendung nicht glauben, die nächste zu sein, die ausgeführt wird, da Faktoren außerhalb Ihrer Kontrolle den Verwalter in seiner Entscheidung beeinflussen können.

`DosSemWait`, `DosSemSetWait` und `DosSemRequest` sind pegelgesteuert. Das bedeutet zum Beispiel, daß ein bei diesen Aufrufen blockierter Thread nur wieder ausgeführt wird, wenn das blockierende Semaphor gelöscht bleibt, bis der blockierte Thread wirklich starten kann. Setzt ein anderer Thread im Computer in der Zwischenzeit das Semaphor wieder, bleibt der blockierte Thread blockiert. Bleibt das Semaphor gelöscht, bis der blockierte Thread wieder an der Reihe ist und starten kann, dann kehrt der Funktionsaufruf zurück, und der ehemals blockierte Thread startet wieder.

`DosMuxSemWait` andererseits ist flankengesteuert. Das bedeutet, daß ein beim Aufruf von `DosMuxSemWait` blockierter Thread zurückkehrt, wenn eines der Semaphore in der Liste gelöscht wird, sogar dann, wenn das Semaphor von einem anderen Thread wieder gesetzt wird, bevor der blockierte Thread erneut starten kann.

```
P()
{
    int blocked=1;

    while( blocked == 1)
    {
        DosSemWait( &countSem); /* warten bis vielleicht ok */

        DosSemRequest( &mutexSem, -1L);   /* ausschließlich */
        if( count == 0)                 /* nicht bereit gesetzt */
        DosSemSet( &countSem);          /* blockieren setzen    */
    else    {
            count--;                    /* Zähler erniedrigen   */
            blocked--;                  /* Schleifenende setzen */
    }
    DosSemClear( &mutexSem);            /* ausschließlich       */
    }
}

V()
{
    DosSemRequest( &mutexSem, -1L); /* ausschließlich      */
    count++;                        /* Zähler erhöhen      */
    DosSemClear( &countSem);        /* Wartende erwecken   */
    DosSemClear( &mutexSem);        /* ausschließlich      */
}
```

Listing 9: *Simulation eines klassischen Semaphors unter OS/2.*

Zusammenfassung

Semaphore in OS/2 können für die Zugriffsteuerung auf geschützte Ressourcen verwendet werden, indem das Semaphor der Ressource zugeordnet wird und die OS/2-Funktionsaufrufe so eingesetzt werden, daß gewährleistet ist, daß das Semaphor zu einer Zeit nur einen Besitzer hat.

Semaphore gestatten es einem Thread auch, einem anderen Thread zu signalisieren, daß ein Ereignis aufgetreten ist. Indem Sie die jeweilige Anwendung analysieren, können Sie das passende Semaphor und die benötigten OS/2-Funktionsaufrufe herausfinden.

Kevin Rudell

Ein Werkzeug setzt sich durch:

TURCK-MESSY+ das PROFI-Maskenentwicklungssystem für viele Programmiersprachen



Vorteile:

- lieferbare Programmierschnittstellen für Turbo C, MS-C, Lattice C, Turbo-Pascal (auch V.4.0), MS-Pascal, MS-Fortran, Modula 2 und Cobol
- mit interaktivem Maskeneditor (what you see is what you get)
- Editieren (ändern) bestehender Masken möglich
- mit Maskenlaufzeitsystem zur Bildschirm- und Tastatursteuerung
- leistungsstarke Debug-Möglichkeiten
- selbstablaufende Demo des Anwenderprogrammes möglich
- zahlreiche Plausibilitätsprüfungen (z.B. Wertebereiche) innerhalb des Maskenlaufzeitsystems möglich (ohne Belastung des Anwenderprogrammes)
- Formeln zwischen Feldern einer Maske möglich (automatisches Rechnen)
- bis zu 8 Windows möglich
- Rapid-Prototyping ohne Programmierkenntnisse möglich
- mit Installationsprogramm für die verschiedenen Rechnertypen (auch nicht »KOMPATIBLE«)
- siehe PC Magazin Nr. 6/1987

Eine komplett lauffähige **Demo-Version** (ohne Programmierschnittstelle) erhalten Sie bei Zahlungseingang von DM 17,50 auf unserem Postgirokonto München 332533-807 (BLZ 70010080). **TURCK-MESSY+** kostet **DM 445,–** (DM 390,– o. MwSt.); jede **Programmierschnittstelle DM 320,–** (DM 280,– o. MwSt.). Mit Handbuch auf 3 Disketten zum Selberausdrucken.

Helmut Turck & Partner GmbH · Wastelbauerstr. 12 c · 8000 München 60 · Tel. 089/8112063

Wichtige englischsprachige Neuerscheinungen für Programmierer:

Programmierwissen im Überfluß

Bei der Programmierung von Personalcomputern ist es für eine schnelle Benutzerschnittstelle fast unumgänglich, direkt auf den Bildschirmspeicher zuzugreifen, denn bei der Programmierung von Bildschirmzugriffen über die BIOS-Routinen wird jedes interaktive Programm unnötig langsam. Wenn man ein solches Programm für nur einen Bildschirmadapter schreibt, ist noch relativ gut an die benötigten Informationen heranzukommen, doch sobald man sein Programm zu einer universellen Version machen möchte, die auf allen gängigen Bildschirmadaptern läuft, steht man im Dunkeln. Die benötigten Informationen über alle Adapter sind fast unmöglich zu beschaffen. Endlich gibt es nun mit »Video Systems« von Richard Wilton ein Werk, das alle Informationen über folgende Bildschirmadapter enthält:

- MDA (Monochrome Device Adapter)
- CGA (Color Graphics Adapter)
- HGC (Hercules Graphics Card & Plus) und Varianten (z.B. ColorCard und InColor Card)
- EGA (Enhanced Graphics Adapter alias HR-Adapter)
- VGA (Video Graphics Array)
- MCGA (Multi-Color Graphics Array)

Doch werden die benötigten Informationen nicht nur aufgelistet, sondern für alle häufig benötigten Zugriffe auch fertige Routinen in Assembler und C vorgestellt, die von C- oder Assemblerprogrammen aufgerufen werden können. Eine Übersicht über die Kapitel des Buches, zeigt sehr deutlich, wie umfassend das Thema behandelt wird.

- Bildschirmhard- und -firmware bei Personalcomputern (Vorstellung der verschiedenen Adapter und des ROM-BIOS)
- Programmierung der Hardware (Funktionelle Bestandteile der Bildschirmadapter, Refreshzyklen, Programmierung des CRT-Controllers, CRTC-Berechnungen, Statusregister, Bildschirmmodi)
- Textmodi (Attribute, Randfarben, Bildschirmflimmern, Puffer, Cursorsteuerung)
- Grafikmodi (Pixeladressierung, Attribute von Pixeln)
- Pixelprogrammierung (Bit-Plane-Programmierung, Pixel lesen und setzen)
- Linien (Linien effektiv zeichnen, Optimierungen, Clipping)
- Kreise und Ellipsen (Algorithmen zum Zeichnen von Ellipsen, Optimierungen, Clipping)
- Bereiche füllen (Füllen mit waagrechten Linien und andere Füllalgorithmen im Vergleich)
- Textanzeige im Grafikmodus (Zeichendefinitionen, Zeichengeneratoren)
- Alphanumerische Zeichensätze (Zeichendefinitionen in RAM und ROM, Veränderung der Zeichendefinitionen, Grafiken im Textmodus)
- Bitblöcke und Animation (Verschiebung von Bitblöcken, Pixeloperationen, Animation, Cursor im Grafikmodus)
- Besonderheiten (Routinen zur Behandlung des Vertikal-Interrupts, Light Pen)
- Grafikroutinen in Hochsprachen
- Übersicht über das Bildschirm-BIOS
- Bildschirm-Hardcopy (Routinen für CGA und EGA)
- Feststellen des Bildschirmadapters (alle Adapter)

Vor allem die Besitzer von Hercules-Adaptern werden es sehr zu schätzen wissen, daß endlich einmal auch dieser Bildschirmadapter in einer seiner Verbreitung würdigen Ausführlichkeit beschrieben wird.

Wer professionelle Software für PCs entwickeln möchte, sei es als Amateur oder Profi, der benötigt dieses Buch. Es ist fast nur Positives darüber zu sagen. Einziger Nachteil des Buchs: Die Listings sind in einem äußerst hellen grün gedruckt, so daß sie nur sehr schwer zu lesen sind. Dies ist umso nachteiliger, da man das Buch wegen seiner Ausführlichkeit und seines Detailreichtums immer wieder zu Hand nehmen und häufig darin lesen wird.

Für PC-Programmierer ist »Video Systems« sicherlich eine der wichtigsten - wenn nicht sogar die wichtigste - Neuerscheinung. Sobald die deutsche Übersetzung, die beim Vieweg-Verlag erscheinen wird, weit genug gediehen ist, werden wir »zum Einlesen« einen Auszug aus diesem sehr interessanten Buch drucken.

Richard Wilton: »The programmer's guide to PC and PS/2 video systems. Maximum performance from the EGA, VGA, HGC, and MCGA«, Redmond: Microsoft Press, 1987; 530 Seiten; ISBN 1-55615-103-9; $24.95. 5¼- oder 3½-Zoll-Diskette zum Buch $25.95 (inkl. Versandkosten).

Probleme mit dem Bildschirmadapter stellen sich für Windows-Programmierer nicht, doch dafür hat Windows einige andere Hürden, die zu nehmen sind, bevor man erfolgreich Windows-Programme schreiben kann. Da sind vor allem der Umfang (fast 500 Funktionen) und auch die gänzlich neue, ereignisgesteuerte Programmierphilosophie von Windows zu nennen. Im *Microsoft System Journal* hat vor allem Charles Petzold mehrmals Licht in das Dunkel der Windows-Programmierung gebracht. Charles Petzold hat nun sein Windows-Programmierwissen in einem 800-Seiten-Wälzer mit dem Titel »Programming Windows« gesammelt. Wer die Artikel von Charles Petzold im *MSJ* gelesen hat, weiß, daß er komplexe Zusammenhänge anschaulich darstellen kann. Das ist ihm in größerem Maßstab auch mit »Programming Windows« geglückt, einem Buch, daß sicherlich für lange Zeit ein Standardwerk für Windows-Programmierer sein wird.

Besonders hervorzuheben sind der didaktische Aufbau des Buchs, die ausführliche Beschreibung der einzelnen Themenkomplexe und die über 60 sehr nützlichen Beispielprogramme, die dabei fast mühelos nebenbei entstehen. Wenn Sie unter Windows programmieren wollen, brauchen Sie dieses Buch.

Es empfiehlt sich, zu diesem Buch auch gleich die Diskette zu bestellen, da die entwickelten Programme und Funktionen keine praxisfernen Beispiel sind, sondern meistens sehr sinnvoll zur Bereicherung eigener Programme verwendet werden können. Einige Programme wurden, meist nicht ganz so ausführlich, bereits im *MSJ* vorgestellt.

- Übergabe von Bildschirminhalten an andere Programme (BlowUp, s. MSJ Nov. '87, S.56)
- Anzeige des freien Speichers (FreeMem in dieser Ausgabe)
- Digitaluhr
- Farbeinstellungen (ColorScr, s. MSJ März '88, S.34)
- Grafiken und Fonts in Menüs
- Dialogboxen zur Dateinamenauswahl
- ein hexadezimaler Taschenrechner (HexCalc, s. MSJ März '88, S.34)
- Abfrage von Systeminformationen
- Grafiken unter Windows
- Grafikanimation
- Schriftauswahl und Anzeige proportionaler Schriften
- Grafikdruck und Bildschirm-Hardcopy
- Laden und speichern der Zwischenablage (Clipboard)
- Verwaltung von Ressourcenbibliotheken

Charles Petzold: »Programming Windows. The Microsoft Guide to programming for the MS-DOS Presentation Manager: Windows 2.0 and Windows/386«, Redmond: Microsoft Press, '1988; 852 Seiten; ISBN 0-914845-91-8; $24.95. 5¼- oder 3½-Zoll-Diskette zum Buch $33.95 (inkl. Versandkosten).

Was Windows-Programmierer an Mehraufwand zu leisten haben, macht sich für Windows-Anwender bezahlt. Doch auch unter Windows gibt es viele Tips, Tricks und Kniffe, die einem den Umgang erleichtern können. Viele dieser Tips hat Jim Heid in seinem Buch »Power Windows« gesammelt und er bietet sie dem Leser übersichtlich eingeteilt und in gut lesbarer Form dar. Besonders Themen wie der Druck auf verschiedenen Ausgabegeräten (speziell Laserdrucker) und die Installation von Softfonts dürften für viele Leser (vor allem im DTP-Bereich) interessant sein. Beschreibungen der wichtigsten Windows-Anwendungen (Excel, PageMaker, Designer, Pro3D uva.) und der Besonderheiten von Windows/386 runden das positive Gesamtbild ab.

Wer Windows häufiger verwendet, sollte sich dieses Buch zulegen, denn er wird viele Dinge effektiver gestalten können. Die zahlreichen detaillierten Beschreibungen von internen Vorgängen und Verhaltensweisen machen das Buch auch für Programmierer interessant.

Jim Heid: »Power Windows. Maximizing the speed and performance of Windows 2.0 & Windows/386«, Redmond: Microsoft Press, 1988; 290 Seiten; ISBN 1-55615-008-3; $19.95.

Geballte OS/2-Informationen

Auch OS/2-Programmierer kommen bei den Neuerscheinungen des Frühjahrs '88 nicht zu kurz. Hier beeindruckt auf den ersten Blick vor allem die Fachkompetenz der Autoren. Für Microsoft Press schreibt Gordon Letwin, der Chefentwickler von Microsoft für OS/2 (»Inside OS/2«), bei Osborne McGraw-Hill ist es Ed Iacobucci, der Leiter des OS/2-Entwicklungsteams bei IBM (»OS/2 Programmer's Guide«). Die Lektüre von »Inside OS/2« ist jedem zu empfehlen, der sich für OS/2 interessiert, für OS/2-Programmierer stellt sie sogar ein unbedingtes Muß dar, denn Gordon Letwin scheut sich nicht, auch über noch nicht angekündigte Zukunftspläne zu reden. Ein Beispiel dafür (»Das Dateisystem von OS/2«) finden Sie im Anschluß an diese Besprechung.

In diesem Buch geht es weniger darum, konkrete Programmbeispiele oder Programmieranweisungen zu geben, sondern mehr darum, einen Überblick über die derzeitigen Möglichkeiten, neuen Konzepte und zukünftigen Entwicklungen von OS/2 zu bieten und darauf hinzuweisen, welche Auswirkungen das auf die Programme und welche Folgen es für Programmierer hat. Für ein tiefgreifendes Verständnis von OS/2 ist das Buch sehr wichtig.

Gordon Letwin: »Inside OS/2«, Redmond: Microsoft Press, 1988; 290 Seiten; ISBN 1-55615-117-9; $19.95.

Ed Iacobucci wendet sich mit seinem »OS/2 Programmers Guide« vor allem an die OS/2-Programmierer, im Besonderen an die Assemblerprogrammierer. Ihm geht es darum, eine möglichst umfassenden Überblick über die Funktionen und Möglichkeiten von OS/2 in der Version 1.0 zu geben.

Assemblerprogrammieren bietet der »OS/2 Programmers Guide« viele Informationen, die so in der OS/2-Dokumentation nicht geboten werden, da diese sehr stark auf die C-Programmierung ausgerichtet ist. Für Programmierer, die andere Sprachen verwenden, ist nur etwa die Häfte des Buchs interessant (wobei das »nur« bei 1100 Seiten sicherlich sehr relativ zu verstehen ist).

Ein großer Nachteil des Buchs ist, daß darin (wie bei IBM üblich) keinerlei Zukunfsaussichten beschrieben werden, selbst auf den Presentation Manager und seine Auswirkungen auf die Programmierung wird nirgends ausführlich eingegangen. Insofern sollte das Buch unbedingt in Kombination mit »Inside OS/2« gelesen werden. Dann ist es durchaus empfehlenswert.

Übrigens sollte man sich nicht von der sehr ähnlichen äußeren Aufmachung des Buchs »Using OS/2« von Kris Jamsa (ebenfalls erschienen bei Osborne McGraw-Hill) täuschen lassen. »Using OS/2« ist der totale Gegensatz von »OS/2 Programmer's Guide«, ein schludrig gemachter »Schnellschuß«, der sein Geld nicht wert ist.

Ed Iacobucci: »OS/2 Programmer's Guide.«, Berkeley: Osborne McGraw-Hill, 1988; 1100 Seiten; ISBN 0-07-881300-X; $24.95. 5¼- oder 3½-Zoll-Diskette zum Buch $24.95.

»Inside OS/2« mit Gordon Letwin, dem OS/2-Chefentwickler:

Das Dateisystem von OS/2

Gordon Letwin ist der Chefentwickler für OS/2 bei Microsoft. Er beschreibt in den folgenden beiden Kapiteln aus seinem Buch »Inside OS/2« das Dateisystem und macht auf zukünftige Entwicklungen und ihren Einfluß auf Anwendungsprogramme aufmerksam.

Dateiverwaltung

Die Dateiverwaltungssysteme von OS/2 und MS-DOS unterscheiden sich voneinander. In OS/2 wird ein einheitliches Schema zur Vergabe von Namen im ASCII-Format für Objekte wie Dateien, Semaphore, gemeinsame Speicherbereiche usw. verwendet. Dieses Konzept wird im folgenden erläutert.

Dateinamen

Bevor die Namensvergabe unter OS/2 erklärt wird, soll das Format der Dateinamen unter MS-DOS angesprochen werden. Unter MS-DOS besitzen Dateinamen das 8.3-Format, mit einem Namensfeld (maximal 8 Zeichen) und einem Erweiterungsfeld (maximal 3 Zeichen)[1]. Der Punkt zwischen dem Namen und der Erweiterung ist nicht Teil des Dateinamens, sondern ein Begrenzungszeichen. Der Dateiname kann ausschließlich Großbuchstaben beinhalten. Bei dem Versuch, einen Dateinamen anzulegen, der Kleinbuchstaben oder eine Mischung aus Klein- und Großbuchstaben enthält, konvertiert MS-DOS den Dateinamen vollständig in Großbuchstaben. Zu lange Dateinamen werden von MS-DOS abgeschnitten. Das Betriebssystem achtet auf die strenge Einhaltung der Regeln und bestimmt dabei auch die Struktur auf der Diskette bzw. Platte. Das Dateiverwaltungssystem, das von MS-DOS Version 3.x unterstützt wird, ist die Dateibelegungstabelle (FAT für file allocation table). Im folgenden finden Sie typische MS-DOS- und OS/2-Dateinamen:

`\FUSSBALL\SRC\KERNEL\SCHED.ASM`

Fussball steht hier stellvertretend für ein Projekt. Der Pfadname führt zum Zeitplan (Scheduler) für das Projekt.

`\MEMOS\286\MODUS.TXT`

In diesem Memo werden die Modi des 80286 behandelt.

`\\HAGAR\ZWISCHEN\GORDON\FUERMARK`

Der Name beschreibt eine Datei in einem Zwischenspeicherverzeichnis auf dem Netzwerk-Server `HAGAR`, die dort vom Benutzer Mark angelegt wurde.

Unter OS/2 sieht die Dateiverwaltung anders aus. Für Mikrocomputer in der Leistungsklasse, die zunehmend verfügbar wird, ist das geschilderte einfache Dateiverwaltungssystem unzureichend. Hier ist beispielsweise an Peripheriegeräte wie WORM-Laufwerke[2] zu denken, die eine spezielle Dateiverwaltung benötigen. Aus diesem Grund ist das Dateiverwaltungssystem in OS/2 nicht fester Bestandteil des Betriebssystems. Vielmehr verfügt OS/2 über ein installierbares Dateiverwaltungssystem (IFS für installable file system). Ein IFS ähnelt einem Einheitentreiber, der entsprechende Programmcode wird beim Start von OS/2 geladen. Die Kommunikation zwischen dem IFS und OS/2 erfolgt über die Standardschnittstelle; das Dateiverwaltungssystem stellt die Software zur Verwaltung des Dateisystems der Speichermedien einschließlich der Möglichkeiten zum Anlegen und Warten von Verzeichnissen, Belegen von Diskettenkapazität usw. bereit.

Sofern Sie mit OS/2 Version 1.0 vertraut sind, wird Ihnen das IFS-Konzept seltsam vorkommen, da es im Handbuch keine Erwähnung findet. Der Grund: Die Implementierung hinkt der architektonischen Konzeption zeitlich hinterher. OS/2 wurde jedoch von Anfang an auf die Unterstützung installierbarer Dateiverwaltungssysteme hin ausgelegt, von denen eines das bekannte FAT-Dateiverwaltungssystem sein kann. Dateiverwaltungssystemaufrufe wie `DosOpen` und `DosClose` wurden vor diesem Hintergrund implementiert. Die strenge Zeitplanung und der daraus resultierende Druck veranlaßten uns, OS/2 Version 1.0 ausschließlich mit einem FAT-Dateisystem ausgestattet auszuliefern – eine künftige Version wird das vollständige IFS-Paket enthalten. Zum Zeitpunkt, da das vorliegende Buch verfaßt wurde, liegt die offizielle Ankündigung von IFS noch in der Zukunft. Informationen hierüber sind aber für die Anwendungsentwicklung unter OS/2 notwendig, daher sollen sie im folgenden gegeben werden.

Da das IFS Datei- und Pfadnamen interpretieren wird und installierbare Dateiverwaltungssysteme große Unterschiede aufweisen können, enthält OS/2 selbst (abgesehen vom IFS) keine Informationen über das Format und die Bedeutung von Datei- und Pfadnamen. Das Datei- und Pfadnamenformat wird vom IFS selbständig verwaltet, sowohl das Anwendungsprogramm als auch OS/2 stellen Zwischenschritte dar, denen keine diesbezüglichen Informationen zur Verfügung gestellt werden müssen. Sie sollten nicht den Versuch unternehmen, Datei- und Pfadnamen zu zerlegen, da das IFS auch Namen in anderen Formaten als dem 8.3-Format von MS-DOS unterstützt. Selbst auf eine feste Länge für einen Datei- oder Pfadnamen kann man sich nicht verlassen. Alle OS/2-Dateinamen- und Pfad-

1 Die Festlegungen bezüglich des Umfangs von Dateinamen gehen auf die Digital Equipment Corporation (DEC) zurück. Die ersten von DEC konstruierten Computer verwendeten die sog. RAD50-Technik zur Speicherung von drei Großbuchstaben in einem 16-Bit-Wort. Daher bot es sich an, im Dateiverwaltungssystem nur Dateinamen aus sechs Zeichen und Namenerweiterungen aus drei Zeichen zu erlauben. CP/M hat diese Dateinamenstruktur übernommen. Da CP/M aber nicht RAD50 als Format verwendete, durfte der Dateiname »großzügigerweise« sogar acht Zeichen umfassen; die drei Zeichen lange Erweiterung blieb erhalten.

2 WORM-Laufwerke (write once read many) enthalten im allgemeinen Laserplatten mit hoher Kapazität. Eine einmal beschriebene Spur kann nicht mehr gelöscht werden. Dennoch läßt sich der Eindruck der Löschbarkeit erwecken, indem neue Versionen von Dateien und Verzeichnissen nach einer Veränderung erneut aufgezeichnet und die alten Versionen abgekoppelt werden.

namenschnittstellen (wie `DosOpen`, `DosFindNext`) wurden auf die Übernahme von Zeichenketten beliebiger Länge ausgelegt. Anwendungsprogramme sollten Namenpuffer mit einer Länge von mindestens 256 Zeichen verwenden, damit sichergestellt ist, daß lange Dateinamen nicht abgeschnitten werden.

Netzwerkzugriff

Eine Länge von 256 Zeichen zur Speicherung eines Dateinamens erscheint auf den ersten Blick etwas übertrieben. Andererseits enthalten OS/2-Dateinamen oft Pfadnamen, die sehr lang werden können. Um einen transparenten Zugriff auf Dateien in einem LAN (local area network) zu erlangen, bindet OS/2 das Netzwerk als Teil des Dateiverwaltungssystems ein. Mit anderen Worten: Ein Pfadname kann neben einem Verzeichnispfad auch einen Gerätenamen enthalten. Ein Anwendungsprogramm könnte einen Öffnungsaufruf mit einem Namenstring wie

```
\ARBEIT\BUCH.DAT
```

oder

```
\\VOGON\TEMP\NEUBER.ASM
```

ausgeben. Der erste Name spezifiziert die Datei `BUCH.DAT` im Verzeichnis `ARBEIT` des aktuellen Laufwerks des lokalen Gerätes. Der zweite Name spezifiziert die Datei `NEUBER.ASM` im Verzeichnis `TEMP` des Gerätes `VOGON`[3]. Künftige Versionen des Microsoft LAN Manager werden das Dateiverwaltungssystem von OS/2 nutzen, so daß Dateinamen (insbesondere programmgenerierte Dateinamen) schnell sehr lang werden können.

Namensgenerierung und Kompatibilität

Anwendungsprogramme sollten idealerweise das Format der vom Benutzer eingegebenen Dateinamen außer acht lassen. Dies ist natürlich unrealistisch. Programme müssen oftmals selbst Dateinamen generieren (zur Zwischenspeicherung von Daten, zur Aufnahme abgeleiteter Dateinamen usw.). Wie kann eine Anwendung Dateinamen anlegen oder verändern und dennoch die Kompatibilität mit allen installierbaren Dateiverwaltungssystemen gewährleisten? Die Antwort: Verwenden Sie den ältesten Standard, von dem Sie sicher annehmen können, das er von allen Systemen unterstützt wird. Ein neues IFS wird FAT-Dateinamen (im 8.3-Format) akzeptieren, da es andernfalls mit vielen anderen Programmen inkompatibel wäre. Wenn ein Anwendungsprogramm sich auf die 8.3-Formatregel beim Anlegen von Dateinamen beschränkt, ist die Kompatibilität mit künftigen Dateiverwaltungssystemen sichergestellt. Im Gegensatz zu MS-DOS schneidet OS/2 (genauer: das Dateiverwaltungssystem von OS/2) den Dateinamen oder die Erweiterung nicht ab, wenn diese zu lang sind; statt dessen wird ein Fehler gemeldet. Die Groß-/Kleinschreibung eines Dateinamens wird auch weiterhin ohne Signifikanz bleiben. Einige Betriebssysteme wie UNIX unterscheiden in Dateinamen zwischen Groß- und Kleinbuchstaben. Die Namen `Arbeit` und `arbeit` bezeichnen in Unix unterschiedliche Dateien. Diese Methode funktioniert, solange ein System durchweg von Programmierern genutzt wird. Für einen Softwareentwickler ist es selbstverständlich, daß es sich bei dem Kleinbuchstaben f (ASCII-Code 66 hex) und dem Großbuchstaben F (ASCII-Code 46 hex) um unterschiedliche Zeichen handelt. Für einen unbedarften Benutzer hingegen bezeichnen f und F den gleichen Buchstaben. Da der Großteil der OS/2-Anwender keine Programmierer sein werden, wird das installierbare Dateiverwaltungssystem von OS/2 keine Unterscheidung bezüglich der Groß-/Kleinschreibung vornehmen.

Wie bereits ausgeführt, sollten programmgenerierte Namen der 8.3-Formatregel folgen, um Komplikationen von vornherein auszuschließen. Ebenfalls aus Sicherheitsgründen empfiehlt es sich, Programmnamen nur derart zu modifizieren, daß alphanumerische gegen andere alphanumerische Zeichen ausgetauscht werden (zum Beispiel: `ARBEIT.OBJ` gegen `ARBEIT.ASM`). Die Verlängerung von Dateinamen ist auch möglich (zum Beispiel `ARBEIT.C` nach `ARBEIT.OBJ`), wenn das Programm den neuen Namen auf seine Gültigkeit hin untersucht. Falls sich der Name als ungültig erweisen sollte, muß eine Korrekturmöglichkeit zur Verfügung stehen. In jedem Fall sollte das Programm erweiterte Dateinamen verarbeiten können. In een oben genannten Substitutionsfällen ist es ratsam, daß der Algorithmus vom Ende der Zeichenkette her arbeitet.

Berechtigungen

In künftigen OS/2-Versionen wird das Dateiverwaltungssystem nicht nur zur Bezeichnung von Dateien verwendet werden können, sondern auch zur Vergabe von Zugriffsberechtigungen. Ein Algorithmus wird eine Zugriffsliste mit jedem Eintrag im Dateinamenbereich verknüpfen, so daß der unautorisierte Zugriff (versehentlich oder vorsätzlich) auf die genannte Datei unterbunden wird.

Andere Objekte im Dateinamenbereich

Die Bedeutung des Dateinamenbereiches wurde bereits anhand mehrerer Aspekte beleuchtet. Zum einen erlaubt der Bereich die Benutzung einer Vielzahl unterschiedlicher Namen. Namen können gruppiert werden (durch das Speichern im gleichen Verzeichnis) und es lassen sich Familien von einheitlichen Namen (durch das Anlegen neuer Unterverzeichnisse) angelegen. Zum zweiten kann der Namenbereich alle Dateien und Einheiten des lokalen Rechners ebenso wie die Dateien und Einheiten von Remote-Systemen umfassen. Schließlich werden Dateisystemnamen künftig einen flexiblen Zugriffsschutz unterstützen.

---

3 Die Benennung von Netzwerken ist komplizierter: Der Name TEMP des Gerätes VOGON bezieht sich auf die verfügbare Netzwerk-Ressource und kann in einem beliebigen aktuellen Verzeichnis stehen.

Es ist keine Überraschung, daß die Entwickler bei der Suche nach Methoden für die Benennung von Objekten, die keine Dateien darstellen (gemeinsam benutzte Speicherbereiche, Systemsemaphore und benannte Pipes), auf den Dateisystemnamenbereich zurückgegriffen haben. Ein Nachteil dieser Entscheidung ist darin zu sehen, daß Dateien und Objekte, die keine Dateien darstellen, unterschiedliche Namen aufweisen müssen. Ein vergebener Dateiname kann nicht mehr für einen gemeinsam genutzten Speicherbereich, ein Systemsemaphor oder eine benannte Pipe verwendet werden, was allerdings verglichen mit den Vorteilen der gemeinsamen Benutzung des Dateisystemnamenbereichs kaum relevant ist. Zudem können für jeden Objekttyp eigene Unterverzeichnisse verwendet werden, so daß Überlappungen von vornherein ausgeschlossen sind.

Bedeutet dies, daß Systemsemaphore, gemeinsam genutzter Speicher und Pipes Dateisystemeinträge auf einer Diskette/Festplatte besitzen? Das FAT-Dateisystem unterstützt in seinen Verzeichnissen keine Sonderobjektnamen. Eine Veränderung in dieser Richtung wäre leicht möglich, doch wäre das FAT-Dateisystem dann nicht mehr abwärtskompatibel (MS-DOS 3.x könnte Disketten/Platten, die unter OS/2 beschrieben wurden, nicht lesen). Da für OS/2 Version 1.0 nur das FAT-Dateiverwaltungssystem verfügbar ist, enthält diese Version RAM-residente Pseudoverzeichnisse, die Objektnamen aufnehmen. Die Namen müssen mit `\SEM\`, `\SHAREMEM\`, `\QUEUES\` und `\DEV\` beginnen, um die Gefahr der Namenskollision mit Dateien in künftigen Versionen von OS/2 zu minimieren.

Obgleich Leistungsmerkmale wie Netzwerkbetrieb und (zukünftig) Zugriffsschutz zum Dateisystemnamenbereich gehören (vom Standpunkt der Architektur her), werden nicht alle Permutationen unterstützt. Insbesondere die Unterstützung von benannten, gemeinsam benutzten Speicherbereichen über ein Netzwerk ist nur mit sehr viel Aufwand zu realisieren.[4] Dieses Leistungsmerkmal wird daher auch künftig in OS/2 nicht implementiert werden.

Das Dateisystem

Das Dateisystem von OS/2-Version 1.0 unterscheidet sich nur geringfügig von dem des MS-DOS-Systems. Dies erklärt sich in erster Linie aus der gewünschten Kompatibilität zu MS-DOS-Programmen. Auf der anderen Seite waren die Zeitvorgaben zur Entwicklung von OS/2 recht knapp. Der Zeitrahmen war nicht ausreichend, um in der ersten OS/2-Version alle gewünschten Leistungsmerkmale unterzubringen, Erweiterungen des Dateisystems sind für künftige Versionen geplant. Im folgenden will ich eine Erklärung liefern, weshalb ein so wichtiger Punkt wie ein neues Dateiverwaltungssystem aufgeschoben wurde.

Die Mikrocomputerbranche entwickelte sich in zwei Richtungen: Massensoftware und Standards. Die »massenhafte« Vermarktung von Software erlaubt den Kauf hochentwickelter Programme für relativ wenig Geld - der Ausdruck wenig Geld muß hier im Vergleich zu den Entwicklungskosten gesehen werden, die oftmals Millionen Dollar betragen. Die Massenvermarktung unterstützt die Standardisierung, da kein Benutzer System, Peripheraeinheiten und Geräte kaufen will, die mit den weithin angebotenen Programmen nicht kompatibel sind. Andererseits unterstützt die Akzeptanz der Standards die Entwicklung von Massensoftware, so daß das Absatzpotential für Programme immens zunimmt. In diesem Kreislauf eröffnet sich die Chance zur Entwicklung äußerst komplexer Anwendungen, da die hohen Gestehungskosten durch einen breiten Massenmarkt abgefangen werden.

Der Synergieeffekt zwischen Standardisierung und Massenvermarktung von Software betrifft natürlich auch die Betriebssystementwicklung. Auf den ersten Blick scheint das Hinzufügen neuer Leistungsmerkmale zu einem Betriebssystem problemlos. Die Entwickler bauen neue Leistungsmerkmale in eine neue Version ein und die Anwendungen werden zur Nutzung der neuen Leistungsmerkmale umgeschrieben. Mit der Massenvermarktung von Software ist dies allerdings nicht möglich. Microsoft könnte eine neue Version von OS/2 mit einigen neuen Leistungsmerkmalen herausbringen (und tatsächlich wird dies auch der Fall sein), anfänglich würden aber nur wenige Anwendungen die neuen Leistungsmerkmale nutzen. Dies beruht auf der begrenzten Marktdurchdringung der neuen Version. Nach einer bestimmten Zeit nach der Verfügbarkeit der neuen Version haben vielleicht zehn Prozent der OS/2-Benutzer ihre Version aktualisiert. Ein ISV (Independent Software Vendor), der ein neues Produkt plant, hofft natürlich auf einen Bestseller. Er muß entscheiden, ob das Produkt die neuen Leistungsmerkmale ansprechen und damit nur zehn Prozent des potentiellen Marktes adressieren soll. Die Alternative besteht in der Verwendung der Leistungsmerkmale der verbreiteten Betriebssystemversion, was das Programm auf allen derzeitigen Systemen ausführbar macht (inklusive der zehn Prozent, die neueste OS/2-Versionen verwenden). Im allgemeinen werden Softwareanbieter die nicht verbreiteten Leistungsmerkmale eines Betriebssytems in ihrem Programm nicht einsetzen, bis die Mehrheit der existierenden Systeme diese unterstützen.

Der Schlüssel zur Einführung neuer Leistungsmerkmale in einem Betriebssystem ist also nicht die Verfügbarkeit oder Nützlichkeit. Der Schlüssel besteht darin, daß die Version mit den neuen Leistungsmerkmalen möglichst rasch eine große Marktdurchdringung erlangt. Falls dies nicht der Fall ist, wird das neue Leistungsmerkmal nicht verwendet. Aus diesem Grunde wurden neue MS-DOS-Versionen nur

---

[4] Dies hängt damit zusammen, daß das gesamte Speichersegment bei jeder Änderung im Netzwerk übertragen werden muß. Durch eine Reihe von Optimierungsmaßnahmen läßt sich der Aufwand reduzieren, was aber immer noch nicht zu einem vernünftigen Aufwand/Nutzen-Verhältnis führt.

herausgegeben, wenn die Unterstützung neuer Hardwareelemente erforderlich wurde. MS-DOS Version 2.0 wurde für die IBM XT-Produktlinie benötigt. MS-DOS Version 3.0 muß der AT-Produktreihe zugerechnet werden. OS/2 stellt keine Ausnahme dar: Das Betriebssystem soll den Protect-Modus der Rechner mit dem Intel-Prozessor 80286 unterstützen. Falls eine neue Version eines Systems nur neue Leistungsmerkmale enthält, die nicht unbedingt gebraucht werden, ist die Marktdurchdringung anfänglich sehr niedrig und verbessert sich nur langsam. Die Kosten zur Aktualisierung sind nur gerechtfertigt, wenn die neuen Leistungsmerkmale unabkömmlich sind. Anwendungsprogramme fordern die Verfügbarkeit neuer Leistungsmerkmale häufig erst, wenn die Verbreitung groß genug ist.

In den Augen der Entwickler ist die erste Version von OS/2 damit sehr wichtig. Durch sie ist eine Möglichkeit zur Einführung neuer Leistungsmerkmale in den Betriebssystemstandard der PCs eingeführt. Dieses »Fenster« wird für lange Zeit nicht mehr so weit geöffnet sein.

Aus diesem Grund wurde die Erneuerung des Dateiverwaltungssystems zurückgestellt: Das Dateiverwaltungssystem kann in späteren Versionen erweitert und verbessert werden, wobei existierende Anwendungen ohne Änderungen weiterhin unterstützt werden. Viele andere OS/2-Leistungsmerkmale mußten dagegen unbedingt in die erste Version des Betriebssystems aufgenommen werden, da sie sonst für Massenanwendungen niemals verfügbar werden würden.

Das OS/2-Dateiverwaltungssystem

Einige kleinere Änderungen wurden am Dateiverwaltungssystem von OS/2 Version 1.0 durchgeführt. Zwei davon sind der Erwähnung wert: Die erste ist die asynchrone Ein- und Ausgabe, die zwei Funktionen umfaßt (`DosReadAsync` und `DosWriteAsync`). Die Funktionen entsprechen den `DosRead`- und `DosWrite`-Aufrufen mit der Ausnahme, daß sie die Kontrolle sofort an das aufrufende Programm zurückgeben (üblicherweise vor der Vervollständigung der Ein-/Ausgabeoperation). Die Funktionen übernehmen die Nummer eines Semaphors, das nach der Beendigung der Ein-/Ausgabeoperation gelöscht wird. Die Threads des aufrufenden Prozesses können das Semaphor zur Überprüfung der Ein-/Ausgabebeendigung verwenden. Die Funktion `DosMuxSemWait` ist in diesem Zusammenhang besonders nützlich, da sie das Warten auf mehrere Semaphorenereignisse erlaubt, die auch asynchrone Ein-/Ausgabe-, IPC- und Zeitgeberereignisse beinhalten dürfen.

Das zweite neue Leistungsmerkmal des Dateiverwaltungssystems ist eine *erweiterte Partitionierung*: Die Unterstützung umfangreicher physikalischer Laufwerke in Form von mehreren Bereichen ist gewährleistet, wobei mehrere Partitionen FAT-Dateiverwaltungssysteme enthalten können. Die Folge ist, daß OS/2 eine große Festplatte als zwei oder mehrere kleinere Einheiten betrachten kann, von denen jede der Größenbeschränkung des Dateiverwaltungssystems unterliegt. Der Aberglaube, daß MS-DOS nur Laufwerke bis zu einer Größe von 32 Mbyte verwenden kann, ist weit verbreitet. Dies ist allerdings nicht richtig. Die Begrenzung liegt darin, daß ein Laufwerk in nicht mehr als 65.535 *Sektoren* aufgeteilt werden kann. Aus der Standardsektorgröße von 512 Bytes ergibt sich der Wert von 32 Mbyte. Hinzu kommt, daß jeder Datenträger auf 32.768 Cluster begrenzt ist. Ein Sektor ist eine Einheit des Diskettenspeicherplatzes; Disketten können immer nur in ganzen Sektoren gelesen und geschrieben werden. Die Größe der Sektoren wird bei der Formatierung des Datenträgers festgelegt. Ein Cluster ist die Einheit, in der die Speicherkapazitätsbelegung für Dateien und Verzeichnisse erfolgt. Ein Cluster kann aus einem oder mehreren vollständigen Sektoren bestehen. Das MS-DOS-Dateiverwaltungssystem erlaubt ein Maximum von 65 Kbyte pro Sektor, während nur 32 Kbyte für Cluster erlaubt sind. Folglich muß eine 32-Mbyte-Platte in Clustern zu zwei (oder mehr) Sektoren belegt werden. Die Entwicklung von Einheitentreibern, die als Sektorgröße ein Mehrfaches von 512 Bytes unterstützen, erlaubt die Umgehung der 65-Kbyte-Sektorbeschränkung und dadurch die Verwendung einer Festplatte, die mehr als 32 Mbyte Kapazität enthält. Der für MS-DOS beschriebene Trick kann auch unter OS/2 angewendet werden, besitzt aber den Nachteil, daß dies die Vergrößerung der Sektoren mit sich bringt, wodurch viel Speicherkapazität verschwendet wird.[5]

Die erweiterte Leistungsfähigkeit zur Partitionierung eines Datenträgers unter OS/2 Version 1.0 stellt eine Zwischenlösung dar, die den Kapazitätsverlust durch die interne Fragmentierung reduziert: Mehr als eine Partition kann ein FAT-Dateiverwaltungssystem enthalten. Partitionierte Datenträger unter MS-DOS dürfen nur eine Partition enthalten, die ein MS-DOS-Dateiverwaltungssystem (das heißt, eine FAT) besitzt. Diese Einschränkung wurde unter OS/2 aufgehoben, so daß beispielsweise eine 60-Mbyte-Festplatte in zwei separate logische Laufwerke partitioniert werden kann (beispielsweise C und D), von denen jede eine Speicherkapazität von 30 Mbyte umfaßt.

Verwaltung der Datenträgernamen

Die Multitasking-Fähigkeit von OS/2 macht eine Erweiterung des Dateiverwaltungssystems im Bereich der Datenträgerkennsätze nötig. Ein Datenträgerkennsatz ist der Name, der der Diskette und den darauf enthaltenen Dateien zugeordnet wird. Ein nichtauswechselbares Laufwerk (Festplatte) enthält stets denselben Datenträger, bei

5 Das FAT-Dateiverwaltungssystem kann mit einem Maximum von 32-Kbyte-Belegungseinheiten (Clustern) arbeiten. Unabhängig von der Größe des Datenträgers müssen alle Dateien eine Mindestdiskettenkapazität von 1/32 Kbyte des Gesamtdatenträgers belegen. Dies bedeutet, daß eine 60-Mbyte-Festplatte unter Verwendung von 1.024-Byte-Sektoren 2.048-Byte-Einheiten belegt.

einem Laufwerk mit austauschbaren Datenträgern (wie Diskettenlaufwerken) kann nicht vorhergesagt werden, welche Diskette vom Benutzer zu welchem Zeitpunkt eingelegt wird.

Die Möglichkeit eines Diskettenwechsels wird in Multitasking-Umgebungen zu einem Problem. Verwendet ein Benutzer beispielsweise eine Textverarbeitung und editiert eine Datei auf der Diskette in Laufwerk A, hat der Editor die Datei geöffnet und behält diesen Status bei der Editierung bei. Ohne das Schließen der Datei oder die Beendigung des Editors kann der Benutzer die Bildschirmgruppe umschalten, damit beispielsweise ein Tabellenkalkulationsprogramm abläuft. In diesem Fall muß in Laufwerk A eine andere Diskette eingelegt werden, die Daten zur Bearbeitung der Tabellenkalkulation enthält. Durch die Umschaltung zum Textverarbeitungsprogramm ohne entsprechenden Diskettenwechsel kommt es für das System zu nicht lösbaren Problemen. Die Betätigung der Taste [PgDn] läßt das Textverarbeitungssystem einen weiteren Sektor der bereits geöffneten Datei lesen. Das Betriebssystem erkennt - aufgrund der FAT-Informationen in den RAM-Puffern - daß der nächste Sektor der Textdatei der Sektor N ist und wird versuchen, den Sektor N der falschen Diskette zu lesen (der Diskette mit den Daten für die Tabellenkalkulation). Der gelesene Sektor wird an das Textverarbeitungsprogramm als nächster Sektor der Textdatei zurückgegeben. Dies ist kein schwerwiegender Fehler; schlimmer wird es aber, wenn das Textverarbeitungssystem einen Sektor auf die Diskette schreibt, wodurch zwei Dateien vernichtet werden: Die Datendatei der Tabellenkalkulation wird durch den nicht korrekt gesteuerten Schreibvorgang zerstört und Gleiches gilt für die Textdatei, bei der beim nächsten Lesezugriff festzustellen ist, daß die auf die falsche Diskette geschriebenen Daten fehlen.

Diese Probleme können nicht dadurch gelöst werden, daß der Benutzer Disketten mit Bedacht einlegt; viele Programme lesen und schreiben ohne direkten Benutzerbefehl auf und von Disketten. So könnte beispielsweise die Textverarbeitung die Arbeit automatisch alle zwei Minuten speichern. Nach Verstreichen dieses Zeitintervalls erfolgt ein vom Benutzer nicht ausdrücklich gewollter Schreibvorgang, auch wenn der Benutzer noch mit der Tabellenkalkulation und der dazugehörigen Tabellenkalulationsdiskette arbeitet.

OS/2 löst diese Probleme, indem Anwendungen ihre Ein-/Ausgaben an geöffnete Dateien nicht an Laufwerk A richten, sondern an einen bestimmten Datenträgerkennsatz (nämlich den, der der Diskette zugeordnet ist, auf der die geöffnete Datei vorliegt). Jeder Datenträgerkennsatz besteht aus einem Datenträgernamen, der im Stammverzeichnis gespeichert ist und einer einzigartigen 32-Bit-Datenträgerkennummer, die sich im Startsektor befindet. In *Bild 1* sind die beiden Teile des Datenträgerkennsatzes gezeigt - einen zur Verwendung durch den Computer (Datenträgerkennummer) und einen für den Bediener (Datenträgername). Jede Dateinummer wird mit einer bestimmten 32-Bit-Datenträgerkennummer assoziiert. Bei einer Ein-/Ausgabeanforderung auf eine Dateinummer überprüft OS/2, ob sich der entsprechende Datenträger im Laufwerk befindet. Dies erfolgt durch den Vergleich des 32-Bit-Wertes der Anforderung mit dem des momentan im Laufwerk befindlichen Datenträgers. Bei übereinstimmenden Nummern wird die Operation vervollständigt. Unterscheiden sich die beiden Nummern, verwendet OS/2 den Hardwarefehler-Daemon-Mechanismus, der eine Aufforderung zum Einlegen des richtigen Datenträgers anzeigt.

Um dies praktikabel zu machen, müssen drei Probleme gelöst werden. Zuerst einmal darf die Überprüfung der Datenträgerkennummer des Datenträgers den Durchsatz nicht mindern. Der Startsektor kann nicht bei jeder Ein-/Ausgabeoperation gelesen werden, wenn dadurch die Ein-/Ausgabegeschwindigkeit um die Hälfte gesenkt wird. Zum zweiten muß sichergestellt sein, daß die Datenträgerkennummer tatsächlich einzigartig ist und drittens muß auch ein Datenträger bearbeitet werden können, der keine Datenträgerkennummern besitzt.

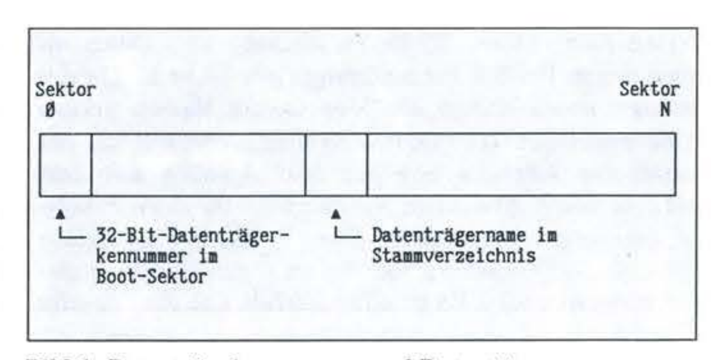


Bild 1: *Datenträgerkennummer und Datenträgername*

Die Überprüfung des korrekten Datenträgers ist einfach, wenn bekannt ist, wann ein Datenträger gewechselt wurde. Es ist offensichtlich, daß die Kennummer nur nach einem Wechsel des Datenträgers gelesen werden sollte. Wie erkennt OS/2 einen Wechsel? OS/2 kann dies nicht erkennen, wohl aber der Einheitentreiber. Falls die Einheit ein nicht austauschbares Medium erhält, muß die Datenträgerkennummer nicht überprüft werden. Der Einheitentreiber erkennt dies und antwortet mit nicht gewechselt, wenn er nach dem Status des Mediums gefragt wird. Einige austauschbare Mediumtreiber besitzen ein Flagbit, das beim Öffnen der Laufwerksklappe gesetzt wird. In diesem Fall antwortet der Einheitentreiber bei einer Anfrage mit Wechselstatus unbestimmt. Der Treiber erkennt nicht, ob der Datenträger tatsächlich ausgetauscht wurde, es könnte aber der Fall sein. In diesem Fall überprüft OS/2 die Datenträgerkennummer. Die erneute Überprüfung der Kennummer wird noch schwieriger, wenn eine Einheit, deren Datenträger austauschbar sind, keinen entsprechenden Indikator besitzt.

Der Entwickler eines Einheitentreibers muß einheitenspezifische Details beachten, um die minimale Zeitspanne zu bestimmen, in der ein Wechsel des Datenträgers auftreten kann. Ist die Einheit für Ein-/Ausgaben bereit und die minimal nötige Zeit zum Austausch des Datenträgers seit der letzten Operation noch nicht vergangen, erkennt der Treiber, daß noch derselbe Datenträger im Laufwerk liegen muß. Ist mehr Zeit vergangen, übergibt der Treiber eine Mediawechsel unbestimmt-Meldung an OS/2 und das Betriebssystem muß den Datenträger überprüfen. Für Diskettenlaufwerke beträgt dieser Zeitintervall üblicherweise zwei Sekunden, so daß nach dem Verstreichen dieses Zeitraumes ein Extralesevorgang ausgeführt wird. Für schnell hintereinander erfolgende Disketteneìn-/-ausgaben werden keine Extralesevorgänge benötigt.

Die Sicherstellung einer einzigartigen Datenträgerkennummer ist ein weiteres Problem. Das Betriebssystem darf sich nicht darauf verlassen, daß der Benutzer keine Datenträgernamen doppelt vergibt. Selbst ein »perfekter Benutzer« kann sich eine Diskette von seinem Nachbarn borgen, die einen Namen trägt, den der Benutzer schon einmal vergeben hat. OS/2 löst dieses Problem durch die Verwendung einer 32-Bit-Zufallszahl als Diskettenkennummer. Bei der Formatierung einer Diskette gibt der Benutzer einen Namen ein. Von diesem Namen werden Prüfsummen gebildet und das Resultat, verknüpft mit der Anzahl der Sekunden zwischen dem aktuellen Zeitpunkt und dem Jahr 1980 wird als Ausgangswert für einen Zufallszahlengenerator verwendet. Der Zufallszahlengenerator gibt eine 32-Bit-Zahl zurück, die als Datenträgerkennummer verwendet wird. Es ist offensichtlich, daß die doppelte Vergabe einer Datenträgerkennummer zwar möglich ist, die vier Milliarden auf diese Weise zu erzeugenden Codes machen dies aber unwahrscheinlich.

Der vom Benutzer eingegebene Name dient lediglich dazu, in späteren Eingabeaufforderungen an den Benutzer klarzumachen, welche Diskette eingelegt werden soll. Der Name muß für das System nicht tatsächlich einzigartig sein. Ein Benutzer, der mehrere Disketten mit dem Namen `Arbeit` verwendet, wird für OS/2 dadurch keine Probleme hervorrufen, da das Betriebssystem die Disketten aufgrund der Datenträgerkennummern als unterschiedlich erkennt. Das Einlegen der falschen Diskette mit Namen `Arbeit` als Reaktion auf eine Eingabeaufforderung wird von OS/2 abgelehnt. Es wird eine erneute Aufforderung zum Einlegen der Diskette ausgegeben. Nachdem das System mehrmals zum Einlegen der Diskette mit dem Namen `Arbeit` aufgefordert hat, wird sich der Benutzer sicherlich zur Umbenennung seiner Disketten entscheiden.

Das schwierigste Problem ergibt sich aus nicht benannten Disketten - dies sind Disketten, die unter MS-DOS formatiert wurden. Es kann auch hier nicht vom Benutzer erwartet werden, daß den Disketten ein einzigartiger Name zugewiesen wird. Gleichfalls ist die automatische Benennung durch OS/2 nicht vertretbar, da Datenträger nur lesbar sein können. Auch wenn dies nicht der Fall ist, ergibt sich das Problem mit Disketten niedriger und hoher Dichte. Disketten niedriger Dichte können in einem Laufwerk für Disketten hoher Dichte gelesen werden, Schreibvorgänge eines mit hoher Dichte arbeitenden Laufwerkes auf eine Diskette mit niedriger Dichte können aber nur von Laufwerken mit hoher Dichte gelesen werden. Das Lesen der neuen Information durch Laufwerke niedriger Dichte ist nicht sichergestellt und der Startsektor ist bei der Verwendung eines Laufwerkes niedriger Dichte unter Umständen nicht mehr lesbar.

Für Datenträger ohne korrekte Datenträgerkennummer versucht OS/2 die Erstellung einer einzigartigen Substitutionsdatenträgerkennummer, indem Prüfsummen von Teilen des Stammverzeichnisses und der FAT-Tabelle gebildet werden. OS/2 verwendet dabei den Datenträgernamen (sofern existent). Ist kein Datenträgername verhanden, versucht OS/2, die Diskette zu beschreiben. Keine der Methoden ist narrensicher und sie erfordern mehrere Diskettenzugriffsoperationen bei der Verifikation eines Datenträgers. Die beste Abhilfe besteht darin, Disketten, die in OS/2-Systemen zum Einsatz gebracht werden sollen, mit Namen zu versehen. OS/2-Namen sind abwärtskompatibel zu MS-DOS Version 3.x.

Der Befehl DISKCOPY erstellt unter OS/2 eine byteidentische Kopie einer Diskette. Die einzige Abweichung der duplizierten Diskette vom Original besteht darin, daß diese eine andere Datenträgerkennummer im Startsektor aufweist (der Datenträgername wird nicht verändert). Der Benutzer von OS/2 erkennt dies nicht, da die DISKCOMP-Utility insofern Falschangaben macht, als das zwei Disketten als identisch angesehen werden, wenn sie bis auf die Datenträgerkennummer übereinstimmen. Wenn ein Benutzer eine Diskette mit DISKCOPY unter OS/2 dupliziert und Original und Kopie mit dem Befehl DISKCOMP unter MS-DOS 3.x vergleicht, wird ein Unterschied festgestellt.

Die Erläuterungen haben sich bis zu dieser Stelle auf Lese- und Schreibvorgänge mit einer geöffneten Nummer beschränkt. Lese- und Schreibvorgänge sind datenträgerorientierte Operationen, da sie sich immer auf den Datenträger beziehen, auf dem die Datei vorliegt. Der `DosOpen`-Aufruf ist dagegen laufwerksorientiert, da das Standard- oder das spezifizierte Laufwerk nach der entsprechenden Datei durchsucht wird, wobei es gleichgültig ist, welcher Datenträger im Laufwerk liegt. Alle Nummernoperationen sind laufwerksorientiert, alle Aufrufe auf Namensbasis sind datenträgerorientiert. Derzeit kann nicht spezifiziert werden, daß eine Datei auf einem bestimmten Datenträger angelegt oder geöffnet wird. Um sicherzustellen, daß eine Zwischen- oder Ausgabedatei auf einem bestimmten Datenträger angelegt wird, muß die Datei auf diesem Datenträger geöffnet sein. Dazu wird direkt vor dem Öffnen ein Schreibvorgang in die Datei ausgeführt. Die Schreiboperation gefolgt von einem `DosBufReset`-Aufruf stellt sicher, daß der gewünschte Datenträger im Laufwerk einliegt.

Effizienz der Ein- und Ausgabe

OS/2 unterstützt blockorientierte Operationen (zerlegend oder nicht zerlegend) bei allen Diskettenein-/-ausgabeoperationen. Ein Programm kann eine beliebige Anzahl von Bytes lesen oder schreiben. OS/2 liest die entsprechenden Sektoren und internen Puffer, so daß nur die betroffenen Bytes geladen werden. Jeder `DosRead`- und `DosWrite`-Aufruf benötigt Ausführungszeit, daher ist es effektiver, mit einem Aufruf eine größere Datenmenge anzusprechen.

Der Durchsatz bei der Ein-/Ausgabe wird nochmals verbessert, wenn eine Sektorgrenzanpassung ausgeführt wird. Das Lesen eines ganzzahligen Vielfachen von 512 Bytes ab einer Sektorengrenze führt dazu, daß vollständige Sektoren ohne Zwischenspeicherung in Systempuffern an die Anzeigehardware übergeben werden. Da das Dateisystem auf eine optimale Speicherung der Daten auf dem Datenträger achtet (logisch zusammenhängende Daten werden so gespeichert, daß das Lesen oder Schreiben ohne umfangreiche Änderung der Schreib-/Lesekopfposition erfolgt), ist das Lesen von vier aufeinanderfolgenden Sektoren (2048 Bytes) nur unwesentlich langsamer als das Lesen eines Sektors (512 Bytes).

Wenn die Länge oder Dateiposition einer Anforderung kein Vielfaches von 512 Bytes ist, führt OS/2 nur die Zerlegung unvollständig zu lesender Sektoren über die Puffer aus. Vollständig zu übertragende Sektoren werden direkt übergeben. Die Übertragung großer Datenmengen mit einer Operation ist auch dann sinnvoll, wenn die Sektorengrenzen bei Anforderungen nicht berücksichtigt werden.

Es läßt sich also feststellen, daß die Effizienz bei der Ein- und Ausgabe am größten ist, wenn umfangreiche Datenmengen angesprochen werden und die Datenanforderungen an den Sektorengrenzen ausgerichtet sind. Auch nicht ausgerichtete Datenanforderungen können mit einem guten Durchsatz ausgeführt werden, wenn umfangreiche Datenmengen in einer Operation bearbeitet werden. Bei Programmen, die keine dieser beiden Voraussetzungen erfüllen, empfiehlt sich die Verwendung der Routinen zum »Blocken« und »Entblocken«. Dies gilt auch bei umfangreichen Datenanforderungen, die nicht an Sektorengrenzen ausgerichtet sind. Programme mit häufigen kleinen Datenanforderungen ohne Sektorenausrichtung arbeiten am besten, wenn Sektorenblöcke in interne Puffer gelesen und anschließend vom Programm selbst entblockt werden. Dies spart den Zeitbedarf für häufige `DosRead`- und `DosWrite`-Aufrufe.

Gordon Letwin

Der Text stammt aus dem Buch Buch »Inside OS/2« von Gordon Letwin, erschienen bei Microsoft Press. Die deutsche Übersetzung wird in Kürze im Vieweg-Verlag erscheinen, mit dessen freundlicher Genehmigung der Abdruck erfolgt.

Impressum

Das *Microsoft System Journal* erscheint alle zwei Monate (ungerade Monatszahlen) etwa Mitte des Vormonats.

Herausgeber, verantwortlich und Anschrift der Redaktion:
Microsoft GmbH, Redaktion *Microsoft System Journal*,
Erdinger Landstr. 2, D-8011 Aschheim-Dornach
Telefon: 089 / 46107-0, Teletex/Telex: (17) 89 83 28, Telefax: 90 63 55
Redaktion:
Günter Jürgensmeier, Haar (jü), Hartmut Niemeier, Wildenberg (ni)
Mitarbeiter dieser Ausgabe:
Kaare Christian, Richard Friedrich, Michael Geary, Allen Holub, Günter Jürgensmeier, Michael Kausch, Hartmut Niemeier, D. Norris, M.J. O'Leary, Charles Petzold, Joachim Schneider, Craig Stinson
Manuskripteinsendungen:
Manuskripte und Programmlistings werden von der Redaktion gerne angenommen. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck und zur Vervielfältigung der Programmlistings auf Datenträgern. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen. Nicht zur Veröffentlichung gelangte Manuskripte und Listings können nur zurückgeschickt werden, wenn Rückporto beiliegt.
Titelgestaltung: Hermann Menig
Anzeigenverkauf: Marianne Nuß
Druck und Abonnements:
schury praxisformulare GmbH, Postfach 270, D-8200 Rosenheim
Bezugspreise: Das Einzelheft kostet DM 19,80. Der Abonnementpreis beträgt DM 115,- für 6 Ausgaben und DM 210,- für 12 Ausgaben. Zu den einzelnen Ausgaben ist zum Preis von DM 19,80 eine Diskette mit allen Listings erhältlich. Das Abonnement inklusive Diskette kostet DM 230,- für 6 Ausgaben und DM 420,- für 12 Ausgaben. In den Preisen enthalten sind Mehrwertsteuer, Versandkosten und Zustellgebühren. Auslandsbezug auf Anfrage. Sollte die Zeitschrift aus Gründen, die nicht vom Herausgeber zu vertreten sind, nicht geliefert werden können, besteht kein Anspruch auf Nachlieferung oder Erstattung vorausbezahlter Bezugsgelder.
Bezugsmöglichkeiten: In Buchhandlungen und im Computer-Fachhandel. Abonnements und Einzelbestellungen: schury GmbH.

Urheberrecht: Alle im Microsoft System Journal erschienenen Beiträge sind urheberrechtlich geschützt. Alle Rechte, auch Übersetzungen, vorbehalten. Reproduktionen gleich welcher Art, ob Fotokopie, Mikrofilm oder Erfassung in Datenverarbeitungsanlagen, nur mit schriftlicher Genehmigung der Microsoft GmbH. Anfragen sind an Michael Kausch zu richten.

Copyright © 1988 Microsoft GmbH. Alle Rechte vorbehalten.

Für die Programme, die als Beispiele veröffentlicht werden, kann der Herausgeber weder Gewähr noch irgendwelche Haftung übernehmen. Aus der Veröffentlichung kann nicht geschlossen werden, daß die beschriebenen Lösungen oder verwendeten Bezeichnungen frei von gewerblichen Schutzrechten sind. Die Erwähnung oder Beurteilung von Produkten stellt, soweit es sich nicht um Microsoft-Produkte handelt, keine irgendwie geartete Empfehlung der Microsoft GmbH dar. Für die mit Namen oder Signatur gekennzeichneten Beiträge übernimmt der Herausgeber lediglich die presserechtliche Verantwortung.

Das *Microsoft System Journal* wird mit Microsoft Word 4.0 geschrieben und gestaltet. Der Ausdruck erfolgt mit den HP-Softfonts AD und AF und dem Programm- und Schriftenpaket *DocuJet* auf einem HP-LaserJet Series II.

Strukturierte Programmierung in BASIC am Beispiel:

Terminalemulation mit QuickBASIC

Der beste Weg, die überragenden Möglichkeiten von QuickBASIC zu demonstrieren, ist ein gutes Programmbeispiel. In diesem Artikel wird mit einem Terminalemulator ein solches Beispiel geliefert.

Das Programm ADM3A.BAS, das in *Listing 1* zu sehen ist, ist ein einfaches Terminalemulationsprogramm. ADM3A macht aus einem PC oder Kompatiblen ein einfaches Bildschirmterminal, in der Art des ADM3A von Lear Siegler. Ich habe gerade diese Terminalemulation gewählt, weil das ADM3A mit den meisten Unix- bzw. Xenix-Systemen und zahlreichen anderen Multiusersystemen verwendet werden kann. Der Emulator ermöglicht den Zugriff auf viele bildschirmorientierte Programme, vom Editor VI bis zu kundenspezifischen Datenbanken.

Um den Emulator in der QuickBASIC-Umgebung zu starten, wird entweder der Start-Befehl im Menü ausgewählt oder im Bearbeitungsmodus einfach [Shift][F5] gedrückt. *Bild 1* zeigt die Auswahl des Befehls `Make EXE File`, mit dem ein ausführbares Programm erzeugt wird, das von DOS aus gestartet werden kann.

Der Programmaufbau

Das Programm ADM3A besteht aus einem einzigen Modul mit einem halben Dutzend Unterprogrammen. Auf der Modulebene werden mit `DECLARE SUB` einige Unterroutinen definiert, damit der Compiler Anzahl und Typ der Parameter überprüfen kann, die an die jeweilige Unterroutine übergeben werden. *Bild 2* zeigt die Modulliste von ADM3A unter QuickBASIC 4.0. In dieser Liste werden Module und Prozeduren innerhalb von Modulen aufgeführt. Auf diesem Bildschirm können Module und Prozeduren ausgewählt, verschoben und gelöscht werden. Der intelligente Editor erlaubt es normalerweise, jeweils eine Programmeinheit zu bearbeiten. Man kann jedoch das Fenster mit dem View-Menü in zwei Teile aufteilen, um verschiedene Teile eines Programms gleichzeitig zu sehen.

Auf der Modulebene werden einige logische Konstanten als Werte, Farben und Tastencodes definiert. Wenn man ein BASIC-Programm liest und auf eine hart codierte Konstante wie zum Beispiel 3 stößt, hat man so gut wie überhaupt keine Information darüber, was dieser Wert bedeutet. Der Zusammenhang kann hilfreich sein.

```
COLOR 3,1
```

wirkt doch etwas schleierhaft, wenn man nicht weiß, was die Zahlen 3 und 1 hier bedeuten. Die Verwendung symbolischer Namen für konstante Werte ist wesentlich besser. So kann mit der Definition

```
CONST BLACK = 0, BLUE = 1, .... , CYAN = 3
```

die `COLOR`-Anweisung wesentlich informativer und übersichtlicher geschrieben werden:

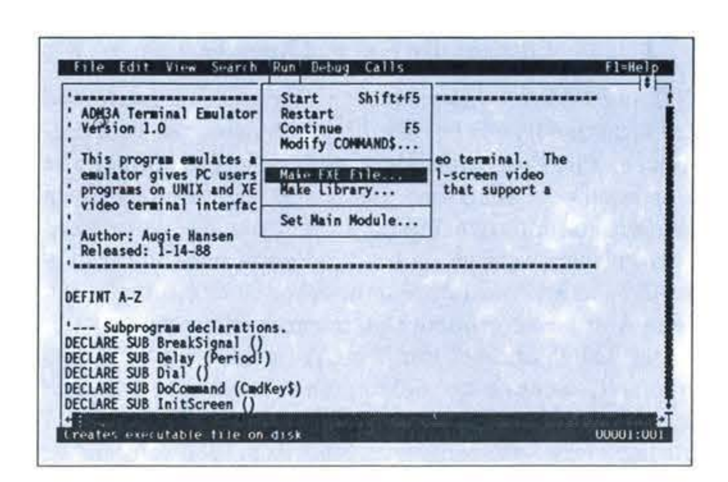


Bild 1: *Die Oberfläche von QuickBASIC 4.0.*

```
COLOR CYAN, BLUE
```

Besonders hinweisen möchte ich auch auf den »RECORD«-Datentyp `WinType`. Diese Fensterart verbindet die Variablen für die Ecken eines Bildschirmfensters (oben, unten, rechts und links). Die Variablen `CmdWin` und `ViewWin` werden mit der `DIM`-Anweisung als Datentyp `WinType` deklariert.

Die Zuweisungen, die auf die Variablendeklaration folgen, richten zwei Fensterbereiche auf dem Programmbildschirm von ADM3A ein. Die oberste Bildschirmzeile ist das Befehlsfenster und der Rest des Bildschirms ist das Anzeigefenster. Diese Aufteilung funktioniert sehr gut, da der Bildschirm eines ADM3A-Bildschirms 24 Zeilen mit 80 Zeichen hat. Da die Befehlszeile oben auf dem Bildschirm plaziert wird, wird der Benutzer beim Wechsel zwischen der Betrachtung des Bildschirms und der Tastatur nicht von etwas anderem abgelenkt. Die Prozedur `InitScreen` kümmert sich um die Aufteilung des Bildschirms in Befehlsfenster und Anzeigefenster. Die Anweisung `VIEW PRINT` bewirkt, daß das Scrollen nur innerhalb des Anzeigefensters auftritt, wenn mit `PRINT`-Befehlen über die unterste Zeile hinaus geschrieben wird. Das Befehlsfenster ist in diesem Programm fest positioniert.

Das Programm ADM3A verwendet eine optionale DOS-Umgebungsvariable für Einstellungen. Wenn in der DOS-Umgebung die Variable `COMPARMS` definiert ist, wird ihr Wert als Parameter in der Anweisung `OPEN COM` verwendet. Ansonsten verwendet ADM3A eingebaute Standardwerte. Es stellt dann die Werte, wie Adresse des Kommunikationsports und die `BREAK`-Signalmaske, die in der Prozedur `BreakSignal` verwendet wird, ein.

Die Hauptschleife überwacht abwechselnd die Tastatur und den Puffer der Schnittstelle auf Eingaben. Aus der Sicht des lokalen Systems nimmt das Programm Eingaben von der Tastatur und schickt sie über die Schnittstelle an das entfernte System. Es liest ankommende Daten von der Schnittstelle und zeigt sie auf dem Bildschirm an. Dieses Grundmuster wird durch einige Sondercodes verändert.

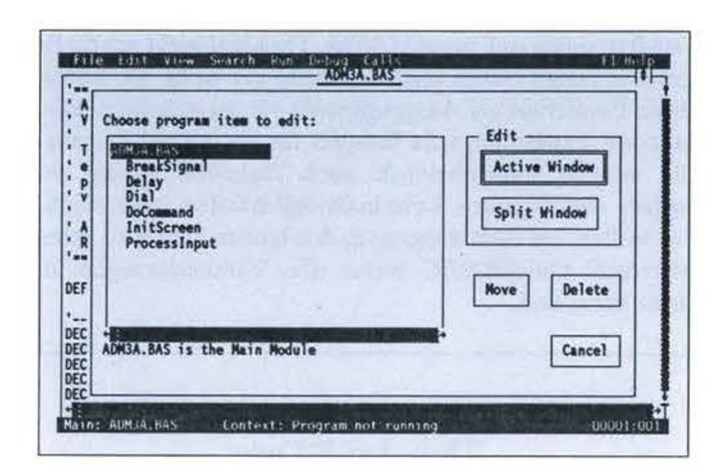


Bild 2: *Die Modulliste von* ADM3A.BAS.

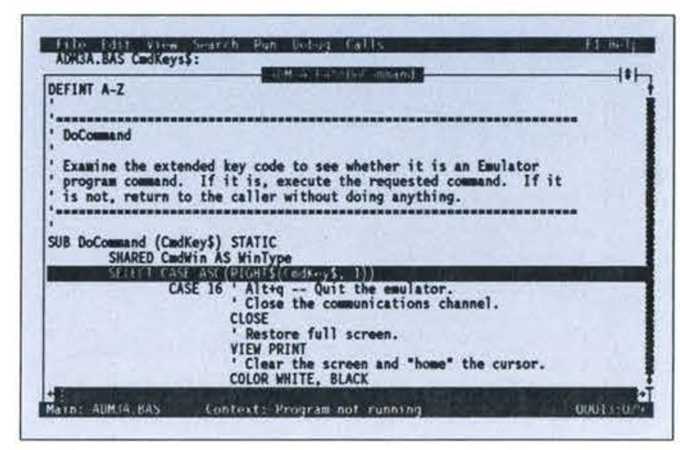


Bild 3: *ADM3A wird mit QuickBASIC 4.0 getestet.*

Wenn der Benutzer irgendetwas auf der Tastatur eingibt, wird es zum entfernten System geschickt, solange es sich dabei nicht um einen erweiterten Code handelt. Die drei ADM3A-Befehle `Break`, `Dial` und `Quit` werden durch erweiterte Codes aufgerufen, die aus Zwei-Byte-Codes bestehen, deren erstes Byte den Wert Null hat.

Wenn `INKEY$` einen erweiterten Code erhält, wird dieser Code als Parameter dem Unterprogramm `DoCommand` übergeben, das den zweiten Code (den Scan-Code) untersucht und ihn mit einer Liste von Befehlscodes vergleicht. Bei den Tastenkombinationen [Alt][B] und [Alt][D] werden andere Prozeduren aufgerufen. Bei dem Befehl [Alt][Q] wird das Emulatorprogramm verlassen, nachdem die Schnittstelle geschlossen, der Bildschirm auf seine volle Größe und normale Attribute eingestellt und der Cursor in die Grundstellung positioniert worden ist. Der vierte Scancode, der von `DoCommand` erkannt wird, ist die Taste [Del]. Bei einem Terminal schickt die Taste [Del] den ASCII-Code 127, der PC jedoch nicht. `DoCommand` wandelt den internen Code also in den vom entfernten System erwarteten um.

Mit der Prozedur `Dial` wird eine einfache Möglichkeit zum Wählen geboten. Wenn der Benutzer [Alt][D] eingibt, antwortet das Programm mit der Eingabeaufforderung »Number:« im Anzeigefenster und wartet darauf, daß eine Telefonnummer eingegeben wird. Die Prozedur `Dial` verwendet dann diese Nummer, um mit dem String »ATDT« vor der Nummer ein Wähl-Kommando an das Modem zu schicken. (Es wird davon ausgegangen, daß Tonwahl bei einem Hayes-kompatiblen Modem verwendet wird.)

Sobald die Verbindung mit dem entfernten System hergestellt ist, muß der Benutzer sich einloggen und die Sitzung so starten, wie es der Hostcomputer erwartet. Während einer Sitzung kann es notwendig sein, dem Host mitzuteilen, daß er mit dem aufhören soll, was er gerade tut. Dies wird im allgemeinen mit einer Break-Taste gemacht, wenn sie vorhanden ist. Unix- und Xenix-Systeme erkennen auch die [Del]-Taste als Unterbrechungstaste.

Die Prozedur `BreakSignal`, die von dem Kommando [Alt][B] aufgerufen wird, schickt ein echtes Break-Signal an den Host. `BreakSignal` arbeitet so, daß es das Break-Bit im Steuerregister der Schnittstelle setzt und es für eine Zeitspanne hochhält, die über die normale Zeitspanne für eine Zeichenübertragung hinausgeht; dann wird das Break-Bit wieder zurückgesetzt.

Die Verzögerung, die hier auf eine halbe Sekunde eingestellt ist, wird von der Prozedur `Delay` erzeugt. Sie verwendet in einer Schleife die BASIC-Funktion `TIMER`, um die Zeit abzuwarten. `Delay` stellt die Startzeit fest, addiert die gewünschte Verzögerungszeit dazu und wartet dann, bis die durch die Summe angegebene Zeit erreicht wird. Um ein »Aufhängen« des Systems zu vermeiden, wenn die PC-Uhr um Mitternacht wieder auf Null zurückgesetzt wird, wird die Prozedur `Delay` abgebrochen, wenn der Zeitwert kleiner als die Startzeit ist.

Die ankommenden Daten werden analysiert. Die meisten Zeichen werden einfach im Anzeigefenster auf dem PC-Bildschirm angezeigt. Einige der empfangenen Zeichencodes haben jedoch eine spezielle Bedeutung: Steuercodes, die eine absolute oder relative Cursorpositionierung bewirken oder den Bildschirm löschen. Das ADM3A verfügt nicht über Funktionen zum Einfügen oder Löschen von Zeilen/Zeichen, die Emulation ist also sehr einfach.

Wenn der Code Escape festgestellt wird, könnte es sein, daß dadurch der Beginn eines Positionierungsbefehls signalisiert wird. Die Cursorposition wird mit Hilfe der Escapesequenz `ESC=zs` gesteuert, wobei `z` die Zeilennummer ist und `c` die Spaltennummer, die beide als ASCII-Werte codiert sind, so daß das Leerzeichen (Code 32) die Zeile bzw. Spalte 0 darstellt. Die Logik des Programms ADM3A entdeckt den Escapecode und wartet dann auf das Gleichheitszeichen. Wenn das folgende Zeichen eines ist, werden zwei weitere Zeichen gelesen und in die entsprechenden Zeilen- und Spaltennummern relativ zum Anzeigefenster (oben und links) umgewandelt und der Cursor entsprechend positioniert.

```
ADM3A EMULATOR                 Break (Alt+b)   Dial (Alt+d)   Quit (Alt+q)
 137  h6|hp2626|hp2626a|hp2626p|2626|2626a|2626p|hp 2626:\
 138          :dc=2\EP:ip=2:se=\E&d@:so=\E&dB:cd=500\EJ:\
 139          :mr=\E&dB:us=\E&dD:mb=\E&dA:mk=\E&dS:me=\E&d@:ue=\E&d@:\
 140          :kh=\Eh:ku=\EA:kl=\ED:kr=\EC:kd=\EB:ks=\E&s1A:ke=\E&s0A:\
 141          :sf=\ES:ta=2^I:xs:tc=2621:
 142  la|adm3a|3a|lsi adm3a:\
 143          :cr=^M:do=^J:nl=^J:bl=^G:\
 144          :am:le=^H:bs:cm=\E=%+ %+ :cl=1^Z:co#80:ho=^^:li#24:ma=^K^P:nd=^L
:up=^K:
 145  cR|c100rv4p|c100 w/4 pages:\
 146          :ti=\EU\Ev  8p\Ep\r:te=\Ev  ~p\Ep\r\n:tc=c100rv:
 147  c4|c1004u|c100 with underline cursor and 4 pages:\
 148          vs=\Ew:ve=\Ew:ti=\EU\Ev  8p:te=\Ev  ~p\Ea7 \E^C:tc=c100uc:
 149  c4|c1004p|c100 w/4 pages:\
 150          :ti=\EU\Ev  8p\Ep\r:te=\Ev  ~p\Ep\r\n:vs@:ve@:tc=c100:
 151  dw|dw2|dw3|decwriter II:\
 152          :cr=^M:do=^J:nl=^J:bl=^G:\
 153          :kb=^h:le=^H:bs:co#132:hc:os:
 154  c8|c1008u|c100 with underline cursor and 8 pages:\
 155          vs=\Ew:ve=\Ew:ti=\EU\Ev  8p:te=\Ev  ^B p\Ea7 \E^C:tc=c100uc:
 156  h4|hp2645|2645|hp45|hp 264x series:\
 157          :mh=\E&dH:mr=\E&dB:us=\E&dD:mb=\E&dA:me=\E&d@:\
 158          :ku=\EA:kd=\EB:kl=\ED:kr=\EC:kh=\Eh:ks=\E&s1A:ke=\E&s0A:tc=hp:
```

Bild 4: *Ein Test von* `ADM3A.BAS` *mit VI unter Xenix.*

Wenn das nächste Zeichen kein »=« ist, druckt das Programm einfach das Escapezeichen und alles was folgt.

Alle anderen Zeichen werden von der Prozedur `ProcessInput` untersucht, um festzustellen, ob sie angezeigt werden können. Einige Codes sind Steuerzeichen, Ctrl-Z löscht zum Beispiel den Bildschirm. Wie beim richtigen ADM3A-Terminal wird darauf geachtet, daß Grenzbedingungen erkannt werden. Bei einer Anforderung den Cursor nach rechts zu bewegen, wenn er bereits am rechten Rand des Fensters ist, wird beispielsweise die `LOACTE`-Anweisung verwendet, um den Cursor zum Anfang der nächsten Zeile zu bewegen, wenn eine vorhanden ist.

Testhilfe durch QuickBASIC

Das Debuggen und der Test des Programms wird durch die Debug-Möglichkeiten von QuickBASIC sehr erleichtert. *Bild 3* zeigt den Bildschirmaufbau, nachdem eine Watch-Ausdruck (`CmdKey$`) und ein Breakpoint auf der Anweisung `SELECT CASE` gesetzt wurden. Wenn das Programm dann gestartet wird, hält es bei dem Breakpoint an, so daß man den Wert des Watch-Ausdrucks untersuchen kann. Es können mehrere Watch-Ausdrücke und Breakpoints verwendet werden. Nach einem Breakpoint wird die Ausführung mit [F5] fortgesetzt.

Die Beispielsitzung, die in *Bild 4* zu sehen ist, zeigt das Programm ADM3A, wie es auf ein Xenix-System zugreift. Es läuft der Editor VI und es wird gerade die Datei `/etc/termcap` bearbeitet. Unter anderem ist der Termcap-Eintrag für das ADM3A-Terminal zu sehen. Jedes andere Programm, das weiß, wie ein ADM3A-Terminal angesteuert wird (was bei den meisten der Fall ist), kann mit dieser Emulation arbeiten.

Eine Grundlage für erweiterte Emulationen

Programme, die »schicke« Liniengrafikzeichen, reverse Darstellung und besondere Zeichenformatierungen verwenden, sehen auf dem ADM3A-Terminal nicht sonderlich gut aus, einige laufen sogar vielleicht gar nicht. Sie können diese Emulation als Ausgangspunkt für ausgefeiltere Emulationen verwenden, zum Beispiel für ein VT100-Terminal. Sie werden wahrscheinlich auch Dateiübertragung und andere »intelligente« Terminalmöglichkeiten implementieren wollen, um dem Programm den letzten Schliff zu geben. Microsoft QuickBASIC bietet alle Voraussetzungen, die dafür nötig sind.

QuickBASIC intern: Threaded-P-Code

Microsoft QuickBASIC 4.0 ist sowohl ein Compiler als auch ein Interpreter. Der Compiler setzt ein Programm in einen Pseudocode um, der zu 90 Prozent aus Maschinencode besteht. Dieser Code wird »Threaded P-Code« genannt (verknüpfter Pseudocode). Der P-Code enthält genug Informationen, um den Quellcode jederzeit rekonstruieren zu können, so daß immer nur eine Version eines Programms im Speicher steht.

Das ist sehr wichtig, da es jederzeit möglich sein soll, ein Programm ohne großen Übersetzungs- und Zeitaufwand zu starten, zu testen und zu bearbeiten. Und weil das Programm nur in einer Form im Speicher steht, und nicht als Quellcode, Objektmodul und ausführbares Programm, steht mehr Speicher für ein Programm zur Verfügung.

Der Threaded-P-Code-Interpreter ist keine separate Programmeinheit, sondern ein verteilter Interpreter. Wenn eine BASIC-Zeile eingegeben wird, prüft QuickBASIC sie sofort auf Syntaxfehler. Wenn Fehler gefunden werden, müssen sie korrigiert werden, bevor eine Fortsetzung möglich ist. Wenn die Syntax in Ordnung ist, wird die Quellzeile in Threaded-P-Code umgesetzt, der aus einer Reihe von Ausführungsadressen besteht. Jede Ausführungsadresse ist einfach nur die Adresse einer Ausführungseinheit.

Eine Ausführungseinheit ist eine Routine, die einen Teil der Arbeit einer BASIC-Anweisung übernimmt. Eine einzelne BASIC-Anweisung erzeugt in der Regel eine Reihe von Ausführungseinheiten, die miteinander verknüpft werden (»threaded«). Die Verknüpfung wird durch einen verteilten Interpreter erreicht, der aus zwei Maschinenbefehlen besteht, die vier Bytes lang sind und sieben CPU-Zyklen benötigen. Jede Ausführungseinheit sieht ungefähr so aus:

`Ausführungsroutine + LODSW ES + JMP AX`

Das ist alles. Ein Programm wird ausgeführt, indem einfach alle Ausführungseinheiten, die für eine bestimmte Aufgabe benötigt werden, miteinander verknüpft werden. Es entsteht kein zusätzlicher Aufwand durch Routinenaufrufe (`CALL/RET`), ein Programm läuft also sehr schnell - etwa so schnell, wie nicht optimierter Maschinencode.

Ein Programm kann sich unter dem P-Code-Interpreter in drei Zuständen befinden: »geparst«, »symbolisch« und

»verknüpft« (threaded). Ein P-Code-System mit nur zwei Zuständen, »geparst« und »verknüpft«, würde sehr langsam sein, da sich die Zeiten für die Umsetzung von geparstem Code in verknüpften P-Code aufsummiert. Durch die Einführung des symbolischen Zwischenzustands hat Microsoft sein Ziel der Möglichkeit für sofortige Programmänderungen erreicht.

Der geparste Zustand ist das Ergebnis der Analyse der BASIC-Anweisungen, entweder von der Tastatur oder aus Dateien. Der größte Teil der Syntaxüberprüfung wird während dieser Übersetzung erledigt. Die Teile eine Programms, die von Bearbeitungen der Anweisungen `COMMON`, `SHARED` oder `DEF` betroffen sind, werden in diesem Zustand gehalten. Die Umsetzung in Threaded-P-Code erfolgt auf einem PC-AT mit einem Durchsatz von 60000 Zeilen pro Minute.

Der symbolische Zustand wird durch effiziente Symboltabellen-Informationen gekennzeichnet, die einen schnellen Datenzugriff ermöglicht. Die meisten Bearbeitungen, die keine globalen Änderungen erfordern, erfolgen in diesem Zustand. Nur die Teile eines Programm, die von den Änderungen berührt werden, müssen in diesen Zustand gebracht werden.

Ein Programm läuft und wird getestet im verknüpften Zustand, in dem es sich die meiste Zeit befindet. Die Umsetzung vom symbolischen in den verknüpften Zustand macht es notwendig, Code zur Typenüberprüfung, Aufrufe von Bibliotheksfunktionen, Kontrollstrukturen und Label auf Speicheradressen einzufügen, sowie das Linken von COMMON-Daten und die Erzeugung der Adressen der P-Code-Ausführungseinheiten.

Wenn nichtglobale Änderungen an einem Programm gemacht werden, werden nur die betroffenen Teile in den symbolischen Zustand zurückversetzt. Bevor das Programm wieder laufen kann, müssen die bearbeiteten Teile wieder in den verknüpften Zustand übersetzt werden. Diese »Rundreise« vom verknüpften in den symbolischen und zurück in den verknüpften Zustand erfolgt mit ungefähr 150000 Zeilen pro Minute. Da man es meistens nur mit einzelnen Programmzeilen oder kleinen Programmteilen wie Funktionen und Unterroutinen zu tun hat, erfolgt die Umsetzung praktisch augenblicklich.

Diese Technologie in QuickBASIC bietet viele Vorteile. Microsoft QuickBASIC weist den Weg in eine Ära erheblich verbesserter Programmiererproduktivität und bringt ein wenig Spaß in den ansonsten eher langweiligen Entwicklungszyklus.

Wie wird es weitergehen? Microsoft-Mitarbeiter haben öffentlich gesagt, daß ein großer Teil der Entwicklungsarbeit für derzeitige und zukünftige Sprachprodukte in die P-Code-Technologie gesteckt wird, die die Grundlage für Compiler-Produkte anderer Sprachen und auch einige Anwendungsprogramme sein wird. Ich hoffe, daß diese Produkte sehr bald angekündigt werden.

Augie Hansen

```
' ADM3A Terminalemulator in QuickBASIC 4.0
' Version 1.0
'
' Dieses Programm emuliert das Terminal Lear Siegler ADM3A.
' Der Emulator erlaubt es PC-Benutzern, bildschirmorientierte
' Programme auf UNIX und XENIX-Systemen laufenzulassen.
'
' Autor: Augie Hansen
' Released: 1-14-88


DEFINT A-Z

'--- Unterprogramm-Deklarationen.
DECLARE SUB BreakSignal ()
DECLARE SUB Delay (Period!)
DECLARE SUB Dial ()
DECLARE SUB DoCommand (CmdKey$)
DECLARE SUB InitScreen ()
DECLARE SUB ProcessInput (Code$)

'--- Konstanten.
CONST TRUE = -1, FALSE = NOT TRUE
CONST BLACK = 0, BLUE = 1, GREEN = 2, CYAN = 3
CONST RED = 4, MAGENTA = 5, BROWN = 6, WHITE = 7
CONST BRIGHT = 8, BLINK = 128
CONST CURSOROFF = 0, CURSORON = 1
CONST BUFSIZE = 512
CONST SPACE = 32
CONST ROWS = 25, COLS = 80
CONST BANNERCOL = 4, COMMANDCOL = 33
CONST TESTMODE = 0

'--- Daten für Bildschirmverwaltung.
TYPE WinType
    Top AS INTEGER
    Left AS INTEGER
    Bottom AS INTEGER
    Right AS INTEGER
    Fgnd AS INTEGER
    Bkgnd AS INTEGER
    Standout AS INTEGER
END TYPE

DIM CmdWin AS WinType
DIM ViewWin AS WinType

CmdWin.Top = 1
CmdWin.Bottom = 1
CmdWin.Left = 1
CmdWin.Right = 80
CmdWin.Fgnd = BLACK
CmdWin.Bkgnd = WHITE
CmdWin.Standout = BROWN + BRIGHT

ViewWin.Top = 2
ViewWin.Bottom = 25
ViewWin.Left = 1
ViewWin.Right = 80
ViewWin.Fgnd = WHITE
ViewWin.Bkgnd = BLUE
ViewWin.Standout = WHITE + BRIGHT

'--- Offsets für Cursorpositionierung.
RowOffset = SPACE - ViewWin.Top
ColOffset = SPACE - ViewWin.Left

'--- Fehlerbehandlung installieren.
ON ERROR GOTO ErrorRecovery

'--- Emulatorbildschirm aufbauen.
InitScreen

'--- Kommunikationsparameter einstellen.
Parm$ = ENVIRON$("COMPARMS")          ' Umgebung prüfen.
IF Parm$ = "" THEN
    Parm$ = "COM2:1200,E,7,1"         ' Standards verwenden.
END IF

Port$ = LEFT$(Parm$, 4)
```

```
IF Port$ = "COM1" THEN
    PortAddress = &H3FB
ELSE
    PortAddress = &H2FB
END IF
BreakMask = &H40                        ' Break-Steuerbits

'--- Schnittstelle öffnen.
OPEN Parm$ FOR RANDOM AS #1 LEN = BUFSIZE

'
' Hauptschleife für Kommunikation.
'
' Tastatur abfragen. Alle normalen eingegebenen Zeichen
' zur Übertragung an das entfernte System über den
' Kommunikationsport schicken. Wenn der Benutzer eine der
' Emulator-Steuertasten drückt, die entsprechende Routine
' aufrufen.
'
Main:
EscapeFlag = FALSE
DO
    '--- Tastatureingaben auf Befehle und Zeichen untersuchen.
    UserKey$ = INKEY$
    IF LEN(UserKey$) > 1 THEN
        DoCommand UserKey$
    ELSEIF UserKey$ <> "" THEN
        '--- Zeichen an entferntes System schicken.
        PRINT #1, UserKey$;
    END IF

    '--- Schnittstelle auf empfangene Zeichen prüfen.
    DO
        IF EOF(1) THEN
            EXIT DO
        END IF
        Received$ = INPUT$(1, #1) ' Ein Zeichen lesen.

        '--- Auf Cursorpositionierungsbefehl prüfen.
        IF EscapeFlag = TRUE THEN
            IF Received$ = "=" THEN
            CursorRow = ASC(INPUT$(1, #1)) - RowOffset
            CursorCol = ASC(INPUT$(1, #1)) - ColOffset
            LOCATE CursorRow, CursorCol
            ELSE
            PRINT CHR$(27); ' Escapecode erhalten.
            PRINT Received$;
            END IF
            EscapeFlag = FALSE
        ELSE
            ProcessInput Received$
        END IF
    LOOP
LOOP

END

ErrorRecovery:
    RESUME Main

'
'
' BreakSignal
'
' Schickt ein Breaksignal an die Schnittstelle.
'
'
SUB BreakSignal
    SHARED PortAddress, BreakMask

    '--- Breakbit setzen.
    OUT PortAddress, (INP(PortAddress) OR BreakMask)

    '--- Verzögerung für Break.
    Delay .5

    '--- Breakbit zurücksetzen.
    OUT PortAddress, (INP(PortAddress) AND NOT BreakMask)
END SUB
```

```
'
'
' Delay
'
' Bewirkt eine Verzögerung. Die Zeitspanne wird als Zahl mit
' einfacher Genauigkeit in Sekunden und maximaler Genauigkeit
' von Zehntelsekunden angegeben.
'
'
SUB Delay (Period!) STATIC
    Start! = TIMER

    '--- Schleife für angegeben Zeitspanne.
    '    Beenden, wenn Uhr "überläuft".
    DO
        Now! = TIMER
        IF (Now! - Start! < Period!) OR (Now! < Start!) THEN
            EXIT SUB
        END IF
    LOOP
END SUB

'
'
' Dial
'
' Benutzer nach Telefonnummer fragen und sie wählen.
'
'
SUB Dial
    INPUT "Number: ", Phone$
    PRINT #1, "ATDT" + Phone$
END SUB

'
'
' DoCommand
'
' Den erweiterten Tastaturcode untersuchen, um festzustellen,
' ob es ein Code für den Emulator ist. Wenn es das ist, den
' geforderten Befehl ausführen. Wenn nicht, zurück zum auf-
' rufenden Programm, ohne etwas zu machen.
'
'
SUB DoCommand (CmdKey$) STATIC
    SHARED CmdWin AS WinType
    SELECT CASE ASC(RIGHT$(CmdKey$, 1))
        CASE 16 ' Alt+q -- Quit: Emulator beenden.
            ' Schnittstelle schließen.
            CLOSE
            ' Ganzen Bildschirm wiederherstellen.
            VIEW PRINT
            ' Bildschirm löschen und Cursor in Grundposition.
            COLOR WHITE, BLACK
            CLS
            LOCATE CmdWin.Top, CmdWin.Left, CURSORON
            END
        CASE 32 ' Alt+d -- Dial: Nummer wählen
            Dial
        CASE 48 ' Alt+b -- Break: Breaksignal schicken.
            BreakSignal
        CASE 83 ' <Del>-Taste -- ASCII DEL schicken.
            PRINT #1, CHR$(127);
        CASE ELSE
            ' Unbekannte Befehle ignorieren.
    END SELECT
END SUB

'
'
' InitScreen
'
' Befehlszeile aufbauen (1 Zeile), sicherstellen, daß der
' Cursor an ist und das aktive Anzeigefenster des Terminals
' einrichten (24 Zeilen).
'
'
SUB InitScreen STATIC
    SHARED CmdWin AS WinType, ViewWin AS WinType
```

```
    '--- Bildschirm auf Text und 80 Spalten einstellen.
    SCREEN TEXTMODE
    WIDTH COLS, ROWS
    COLOR WHITE, BLACK
    CLS

    '--- Befehlsfenster in oberster Zeile anzeigen.
    LOCATE CmdWin.Top, CmdWin.Left, CURSORON
    COLOR CmdWin.Fgnd, CmdWin.Bkgnd
    PRINT SPACE$(CmdWin.Right - CmdWin.Left + 1)

    '--- Programmnamen anzeigen.
    LOCATE CmdWin.Top, CmdWin.Left + BANNERCOL
    COLOR CmdWin.Standout, CmdWin.Bkgnd
    PRINT "ADM3A EMULATOR";

    '--- Befehle anzeigen.
    LOCATE CmdWin.Top, CmdWin.Left + COMMANDCOL
    COLOR CmdWin.Fgnd, CmdWin.Bkgnd
    PRINT "Break (Alt+b)   Dial (Alt+d)   Quit (Alt+q)"

    '--- Terminalbildschirm anzeigen.
    VIEW PRINT ViewWin.Top TO ViewWin.Bottom
    COLOR ViewWin.Fgnd, ViewWin.Bkgnd
    CLS
END SUB

'
'
' ProcessInput
'
' Input von der Schnittstelle analysieren. Auf ADM3A-Befehle
' reagieren. Alles andere unverändert im Anzeigefenster
' ausgeben.
'
'

SUB ProcessInput (Code$) STATIC
    SHARED EscapeFlag, ViewWin AS WinType
    SELECT CASE ASC(Code$)
        CASE 8 ' ASCII Backspace-Zeichen
            IF POS(0) > ViewWin.Left THEN
                ' nichtdestruktives Backspace
                LOCATE , POS(0) - 1
            END IF
        CASE 10 ' ^J -- Newline-Zeichen
            IF CSRLIN < ViewWin.Bottom THEN
                LOCATE CSRLIN + 1
            ELSE
                PRINT Code$;
            END IF
        CASE 11 ' ^K -- Zeile nach oben
            IF CSRLIN > ViewWin.Top THEN
                LOCATE CSRLIN - 1
            END IF
        CASE 12 ' ^L -- Formfeed-Zeichen
            ' ADM3A verwendet nichtdestruktives Leerzeichen
            IF POS(0) < ViewWin.Right THEN
                LOCATE , POS(0) + 1
            ELSEIF (POS(0) = ViewWin.Right) AND
                        (CSRLIN < ViewWin.Bottom) THEN
                LOCATE CSRLIN + 1, ViewWin.Left
            END IF
        CASE 13
            LOCATE , ViewWin.Left
        CASE 26 ' ^Z -- Bildschirm löschen
            CLS
        CASE 27 ' Esc -- ev. Beginn einer Escapesequenz
            EscapeFlag = TRUE
        CASE 30 ' ^^ -- Cursor in Grundposition
            LOCATE ViewWin.Top, ViewWin.Left
        CASE ELSE
            PRINT Code$;
    END SELECT
END SUB
```

Listing 1: *Der Quellcode der ADM3A-Terminalemulation in QuickBASIC 4.0.*

Reservierte Anwendungsnamen

F: *Wir haben eine Windows-Anwendung mit dem Namen* DISPLAY.EXE. *Windows kann sie nicht aufrufen. Wenn wir die Datei auf so ziemlich jeden beliebigen anderen Namen umbenennen, kann die Anwendung gestartet werden. Haben wir gegen irgendwelche Regeln verstoßen oder einen reservierten Namen verwendet? Wenn ja, beschreiben Sie bitte alle diese Regeln.*

A: Sie haben einen Namen verwendet, der nicht für eine Anwendung verwendet werden darf. DISPLAY.EXE ist für die Hersteller von Bildschirmadaptern für die Erstellung von Gerätetreibern reserviert, damit sie diese testen können. Im folgenden werden einige weitere Namen aufgelistet, die Sie nicht verwenden dürfen:

SPOOLER.EXE	MOUSE.EXE
CLIPBRD.EXE	SYSTEM.EXE
CONTROL.EXE	WINOLDAP.EXE
GDI.EXE	KEYBOARD.EXE
USER.EXE	COMM.EXE
KERNEL.EXE	WIN.EXE
MSDOS.EXE	WIN200.EXE
MSDOSD.EXE	

Profi-Tools für QuickBASIC

Schreiben Sie schnellere, leistungsfähigere und professionellere Programme! Wir helfen Ihnen dabei mit nützlichen Tools.
Zum Beispiel:
- **Toolboxen (Fenstertechnik, Menüs, DOS-Funktionen etc.)**
- **Relationale Datenbank mit komfortablem Masken-Editor**
- **Grafik-Paket (Geschäftsgrafik und Zeichensatz-Generator)**
- **Maus-Unterstützung für Ihre Programme**

Alle Pakete mit ausführlich dokumentierten Quelltexten und Programmbeispielen. Wo erforderlich, kommen schnelle Assembler-Routinen zum Einsatz. Wollen Sie mehr aus Ihrem BASIC-Compiler herausholen? Wir informieren Sie gerne kostenlos!

Ingenieur-Büro Harald Zoschke
Berliner Str. 3, D-2306 Schönberg/Holstein
Telefon 04344/6166
Eingetr. Warenzeichen: QuickBASIC: Microsoft;

Herausfinden, wo sich Windows befindet

F: *Wie kann ich den Pfad des Windows-Verzeichnisses aus einer Anwendung heraus feststellen?*

A: Verwenden Sie `GetModuleFileName` und `GetModuleHandle`. Im folgenden ein Beispiel dafür, wie man feststellen kann, von wo aus Windows geladen wurde:

```
#define PATHMAX 80

char szTemp[PATHMAX+1];

GetModuleFileName
      ( GetModuleHandle((LPSTR)"kernel"),
        szTemp, PATHMAX );

/* szTemp enthält nun Laufwerk und Pfad
   von/aus dem Windows geladen wurde  */
```

Ein Fenster durch ein anderes ersetzen

F: *Der MS-DOS Executive kann ein Fenster durch ein anderes ersetzen, wie macht er das?*

A: Wenn das höherwertige Byte des Flag-Wortes, bei ShowWindows (!= 0xFF) ist, wird es mit dem anfordernden Fenster ausgetauscht. Verwenden Sie diese Möglichkeit auf eigene Verantwortung, die Schnittstelle kann jederzeit ohne spezielle Mitteilung geändert werden.

MANCHE SIND GANZ SCHÖN ÜBERSPOILERT.



ECHT SCHNELL DAGEGEN: MICROSOFT QUICKBASIC 4.0.



Eine Schnecke bleibt eine Schnecke – auch wenn sie sich durch allerlei Mimikry – Spoiler, wohlklingende Namen und ähnliches – ein rasantes Outfit schneidert.

Werfen Sie dagegen bei der neuen deutschen Version von Microsoft QuickBASIC 4.0 mal einen Blick unter die Haube: Da gibt es statt Show-Tuning einen völlig neu überarbeiteten Compiler. So sensationell schnell, daß Sie damit wie mit einem Interpreter arbeiten können: Programm ausführen, anhalten zum Debuggen und Verändern, einfach weiterlaufen lassen – ohne lästige Wartezeit. Programmänderungen baut Microsoft QuickBASIC 4.0 um die 150.000 Zeilen pro Minute ein – eine echte Revolution! Revolutionär auch die automatische Syntaxüberprüfung direkt beim Eintippen des Programmcodes. Fehler werden sofort angezeigt – die integrierte Hilfefunktion liefert dazu schnellstens Antworten.

Ein Compiler, so schnell, daß Sie wie mit einem Interpreter arbeiten. Integrierte Entwicklungsumgebung: Compiler, Editor und Debugger in einem. Syntaxüberprüfung bei der Eingabe und kontextsensitive Hilfe.

Und hier das absolut neue Fahrgefühl: Aufrufe der Microsoft Sprachen C 5.0, QuickC, FORTRAN 4.0, Makroassembler und PASCAL 4.0 werden ebenso unterstützt wie die Herkules Graphikkarte und die Intel 8087/80287 Koprozessoren.

Also umsteigen, bei uns einsteigen und ab geht die Post. Eine Probefahrt wird Sie restlos überzeugen.

MS/DOS | 320/KB | 3½ 5¼



ZUKUNFT DER SOFTWARE

COUPON

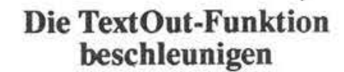
Bitte senden Sie mir Informationsmaterial zu Microsoft QuickBASIC 4.0.

Ich nutze Software: ☐ privat ☐ beruflich/Branche ____________

Mein Rechner: ☐ MS-DOS ☐ MS-OS/2 ☐ Macintosh

Bitte senden Sie den Coupon an: Microsoft GmbH · Erdinger Landstraße 2 · 8011 Aschheim-Dornach

Absender nicht vergessen.



Die TextOut-Funktion beschleunigen

F: *Gibt es eine Möglichkeit, die Funktion* `TextOut` *von Windows zu beschleunigen?*

A: Ja, um die Geschwindigkeit der Funktion `TextOut` zu beschleunigen, verwenden Sie folgende Methode:

1. Berechnen Sie die obere linke Ecke der ersten Zeichenzelle von `TextOut` in Client-Koordinaten.
2. Wandeln Sie die Client-Koordinaten in Bildschirmkoordinaten um (`ClientToScreen`, physische Einheiten).
3. Richten Sie die Bildschirmkoordinaten mit Moduloarithmetik auf eine 8-Bit-Grenze aus (/ oder %). Der Bildschirm wird Zeile für Zeile verwaltet und ist auf Worte ausgerichtet. Durch die Ausrichtung auf die Bytegrenze wird die Rotierung von Bits bei `TextOut` vermieden.
4. Wandeln Sie die ausgerichteten Bildschirmkoordinaten in Client-Koordinaten zurück.

Ein Piepser unter Windows

F: *Was ist die einfachste Methode, unter Windows einen Piepser zu erzeugen (wie bei MessageBeep, jedoch ohne Meldungsbox)?*

A: Verwenden Sie `MessageBeep()` mit jeder beliebigen Konstante außer denen, die in `WINDOWS.H.` definiert sind. Die Konstante ist die relative Dauer des Piepsers. Es wird keine Meldungsbox angezeigt. Diese Möglichkeit ist nicht dokumentiert.

Termine ... Termine ... Termine ... Termine

Mit Microsoft-Seminaren sicher in die Zukunft

Das Betriebssystem der Zukunft heißt Microsoft OS/2. Microsoft Windows und der Presentation Manager sind bzw. werden die Benutzeroberflächen der Zukunft sein. Für professionelle Entwickler bedeutet das, sich ab sofort mit dieser neuen Software auseinandersetzen zu müssen. Damit schaffen sie die Voraussetzung, schnellstmöglich Programme in der »neuen Welt« verfügbar zu haben.

Natürlich wird die Umstellung auf das neue Betriebssystem sowie die Programmerstellung nicht von heute auf morgen vollzogen sein. Um den Anfang jedoch so einfach wie möglich zu gestalten, bietet Microsoft eine neue Dienstleistung an: Das Microsoft Institut.

Die Spezialseminare des Microsoft Instituts vermitteln in kleinen Gruppen intensiv all das, was zum Einstieg in die Programmentwicklung nötig ist. Modernste Trainingsmethoden sowie PC-Demonstrationen und -Übungen sind selbstverständlich. Die Dozenten befassen sich auch im persönlichen Gespräch ausführlich mit den individuellen Forderungen und Problemen der Teilnehmer. So bekommen professionelle Entwickler durch professionelle Schulung die Möglichkeit, ihren hohen Wissenstand den neuen Gegebenheiten anzupassen.

Jeder Interessent in der Bundesrepublik Deutschland, der Schweiz und Österreich hat die Chance, sich mit der neuen Welt von Microsoft OS/2 und Microsoft Windows auseinanderzusetzen. Denn das Microsoft Institut arbeitet vor Ort mit kompetenten Schulungsunternehmen zusammen:

Digicomp AG, Zürich
Elektro-Calcul PI S.A., Lausanne
Integrata GmbH, Tübingen
INTEL Semiconductor GmbH, München
Olivetti Bildungs-Zentrum GmbH, Düsseldorf
Ueberreuter Media GmbH, Wien.

Die Dozenten werden speziell von Microsoft ausgebildet und stehen in ständigem Kontakt mit uns. So gibt es keine Informationsverluste: Das Wissen wird immer aktuell und aus erster Hand vermittelt. Microsoft erstellt die Seminare und die Seminarunterlagen und gewährleistet Qualität durch die Auswertung der Seminare.

Das Microsoft OS/2 Einführungs-Seminar

Das zweitägige Seminar wendet sich an PC-Software-Entwickler, die Programmierkenntnisse in einer höheren Programmiersprache wie C, Pascal, o.ä. besitzen.

Die Teilnehmer lernen im Vortrag und in Diskussionen das Konzept von Microsoft OS/2 kennen und erhalten einen Überblick über die Fähigkeiten und Programmierschnittstellen dieses Betriebssystems. Während des Seminars haben die Teilnehmer die Möglichkeit, das Gelernte anhand von Übungsaufgaben für sich selbst zu überprüfen.

Ort	Datum	Veranstalter
Düsseldorf	04./05.07.	OBZ
	01./02.08.	OBZ
	05./06.09.	OBZ
Frankfurt	21./22.07.	Integrata
	05./06.09.	Integrata
Hamburg	07./08.07.	Integrata
	08./09.08.	Integrata
München	11./12.07.	Intel
	18./19.07.	Integrata
	18./19.07.	OBZ
	01./02.08.	Integrata
	22./23.08.	OBZ
	29./30.08.	Integrata
	12./13.09.	Intel
	15./16.09.	Integrata
Münster	19./20.09.	Integrata
Tübingen	04./05.07.	Integrata
	25./26.07.	Integrata
	22./23.08.	Integrata
	26./27.09.	Integrata
Graz	15./16.09.	Ueberreuter
Innsbruck	19./20.09.	Ueberreuter
Wien	11./12.07.	Ueberreuter
	29./30.08.	Ueberreuter
	15./16.09.	Ueberreuter
	19./20.09.	Ueberreuter
Genf	14./15.09.	Electro Calcul
Zürich	07./08.07.	Digicomp
	22./23.08.	Digicomp
	21./22.09.	Digicomp

Der Microsoft OS/2 Workshop

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrungen in einer höheren, strukturierten Programmiersprache unter MS-DOS und C-Kenntnisse besitzen sowie das MS-OS/2-Einführungsseminar besucht haben.

Die Teilnehmer lernen im Vortrag und praktischen Übungen am PC Family-API-Programme zu schreiben und Device-I/O-Routinen zu erstellen sowie Multitasking-Funktionen zu nutzen und eigene Dynamic-Link-Bibliotheken zu erstellen; außerdem können sie die erweiterten Speicherverwaltungsmöglichkeiten des Intel 80286 nutzen und mit Hilfe des MS-OS/2 Memory Managers programmieren. Dieses Seminar ist übrigens nicht im SDK-Preis enthalten.

Termine ... Termine ... Termine ... Termine

Ort	Datum	Veranstalter
Düsseldorf	06./07./08.07.	OBZ
	03./04./05.08.	OBZ
	07./08./09.09.	OBZ
Frankfurt	07./08./09.09.	Integrata
Hamburg	10./11./12.08.	Integrata
München	13./14./15.07.	Intel
	20./21./22.07.	OBZ
	03./04./05.08.	Integrata
	31./01./02.09.	Integrata
	14./15./16.09.	Intel
	21./22./23.09.	OBZ
Münster	21./22./23.09.	Integrata
Tübingen	27./28./29.07.	Integrata
	28./29./30.09.	Integrata
Genf	26./27./28.09.	Electro Calcul
Wien	05./06./07.09.	Ueberreuter
Zürich	13./14./15.07.	Digicomp
	29./30./31.08.	Digicomp

Das Microsoft Windows Einführungs-Seminar

Das zweitägige Seminar wendet sich an PC-Software-Entwickler, die Programmierkenntnisse in einer höheren Programmiersprache wie C, Pascal, o.ä. besitzen.

Die Teilnehmer lernen im Vortrag und in Diskussionen das Konzept von Microsoft Windows kennen und erhalten einen Überblick über dessen Fähigkeiten und Programmierschnittstellen. Dieses Seminar ist nicht im SDK-Preis enthalten.

Ort	Datum	Veranstalter
Düsseldorf	11./12.07.	OBZ
	15./16.08.	OBZ
	12./13.09.	OBZ
Frankfurt	01./02.08.	Integrata
	22./23.08.	Integrata
Hamburg	11./12.07	Integrata
	12./13.09.	Integrata
München	18./19.07.	Integrata
	25./26.07.	Intel
	25./26.07.	Integrata
	01./02.09.	Integrata
	26./27.09.	Integrata
Münster	08./09.08.	Integrata
Tübingen	07./08.07.	Integrata
	15./16.08.	Integrata
	29./30.08.	Integrata
	19./20.09.	Integrata
Graz	26./27.09.	Ueberreuter
Innsbruck	05./06.09.	Ueberreuter
Wien	22./23.08.	Ueberreuter
Zürich	11./12.08.	Digicomp
	21./22.09.	Digicomp

Der Microsoft Windows Workshop

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrungen in einer höheren, strukturierten Programmiersprache unter MS-DOS und C-Kenntnisse besitzen sowie das Microsoft Windows Einführungsseminar besucht haben.

Die Teilnehmer lernen im Vortrag und praktischen Übungen am PC Benutzerschnittstellen zu erstellen, die grafische Programmierschnittstelle zu nutzen, die Routinen zum Memory Management anzuwenden und dynamische Bibliotheken zu erstellen und zu benutzen. Dieses Seminar ist nicht im SDK-Preis enthalten.

Ort	Datum	Veranstalter
Düsseldorf	13./14./15.07.	OBZ
	17./18./19.08.	OBZ
	14./15./16.09.	OBZ
Frankfurt	24./25./26.08.	Integrata
Hamburg	14./15./16.09.	Integrata
München	27./28./29.07.	Integrata
	27./28./29.07.	Intel
	17./18./19.08.	Integrata
Münster	10./11./12.08.	Integrata
Tübingen	21./22./23.09.	Integrata
Wien	24./25./26.08.	Ueberreuter
Zürich	10./11./12.08.	Digicomp
	26./27./28.09.	Digicomp

Microsoft Excel für Programmierer

Das dreitägige Seminar wendet sich an PC-Software-Entwickler, die Programmiererfahrung in einer höheren Programmiersprache haben.

Die Teilnehmer lernen im Vortrag und in praktischen Übungen am PC das Konzept und die Möglichkeiten, mit Microsoft Excel-Makros Applikationen zu erstellen.

Ort	Datum	Veranstalter
Düsseldorf	18./19./20.07.	OBZ
	08./09./10.08.	OBZ
	19./20./21.09.	OBZ
Hamburg	10./11./12.08.	Integrata
Tübingen	13./14./15.07.	Integrata
Baden	12./13./14.09.	Integrata
Wien	16./17./18.08.	Ueberreuter
	12./13./14.09.	Ueberreuter
Neuchatèl	29./30./31.08.	Electro Calcul

Tips zur C-Programmierung:

Dynamische Speicherverwaltungstechniken

Die meisten Anwender besitzen zwischen 64 und 640 Kilobyte Arbeitsspeicher in Ihren PCs, und es kann fast davon ausgegangen werden, daß viele ihre Computer noch mit Zusatzspeicher erweitert haben. Obwohl wir die Ober- und die Untergrenzen der Speicherausbauten kennen, können wir nie ganz sicher sein, wieviel Speicher der einzelne Computer besitzt.

Programme können geschrieben werden, um in einer fest definierten Umgebung zu laufen, oder sie können sich dynamisch an den verfügbaren Speicher angleichen. Programme, die eine flexible Speicherverwaltung unterstützen, sind besonders nützlich. Ich merkte dies, als meine Lieblingstextverarbeitung die gerade zusätzlich installierten 128 Kbyte Arbeitsspeicher nicht nutzen konnte. Stattdessen griff sie weiter ständig auf das Diskettenlaufwerk des PCs zu.

Hätten die Programmierer, die mein Textverarbeitungssystem geschrieben haben, von dynamischer Speicherverwaltung Gebrauch gemacht, könnte das Programm den zusätzlichen Speicher für größere Textteile oder sogar für einen größeren Teil des Programms selbst verwenden.

Dynamisches Allokieren erlaubt es Programmen, sich selbst an die Umgebung anzupassen und damit die Vorteile des verfügbaren Speichers zu nutzen. Solche Programme allokieren Speicher dynamisch, um Arrays und verkettete Listen zu verwalten, anstatt feste Arraygrößen zu verwenden, die der Programmierer bei der Deklaration festlegen muß. Die einzige Grenze der Arraygröße ist der verfügbare Speicher auf dem System des Anwenders.

Dieser Artikel erläutert den Einsatz der C-Standardfunktionen, um Programme zu schreiben, die Datenstrukturen verwenden, die dynamisch wachsen und kleiner werden, während das Programm läuft. Wir implementieren Funktionen für Arrays und verkettete Listen, die für dynamische Speicherverwaltung besonders gut geeignet sind.

Die Standardfunktionen `malloc` und `free` gehören zu den Werkzeugen, die hier erläutert werden. Die Funktion `malloc` erwartet als Argument die Anzahl der zu allokierenden Byte und gibt als Ergebnis einen Zeiger auf den Anfang des Speicherblocks zurück, vorausgesetzt, es ist noch genügend Speicher vorhanden. Ein Aufruf der Funktion `free` gibt den dynamisch allokierten Speicher an den Pool des verfügbaren Speichers zurück.

Das Problem

Wie die meisten Programmiersprachen, so verlangt auch C, daß Sie bei der Deklaration eines Arrays seine Größe festlegen. Dies erweist sich häufig als Einschränkung für ein Programm, da eine Obergrenze für die Menge der zu bearbeitenden Daten gesetzt wird. Wieso nicht das Array so groß machen, daß die größtmögliche Datenmenge bearbeitet werden kann?

```
main()
{
    short       zaehler;                /* 2 Byte Speicher */
    static char namen_liste[1000][25] = {'\0'};
                                        /* 25000 Byte */
    int         status_vektor[10];     /* 20 Byte bei 16 Bit CPU */
                                        /* 40 Byte bei 32 Bit CPU */

    printf(
    "sizeof zaehler %d, namen_liste %d, status_vektor %d\n",
            sizeof zaehler, sizeof namen_liste,
            sizeof status_vektor);

    printf("int's sind %d Bytes, Ich bin eine %d Bit-CPU\n",
            sizeof(int), 8 * sizeof(int));
}
```

Die Ausgabe des obigen Programms:

```
sizeof zaehler 2, namen_liste 25000, status_vektor 20
int's sind 2 Bytes, Ich bin eine 16 Bit-CPU
```

Listing 1: *Die Verwendung von* `sizeof` *zur Feststellung des Speicherbedarfs.*

Unglücklicherweise ist der Speicher nicht grenzenlos. Je mehr Speicher ein Programm benötigt, um so größer muß das System sein, auf dem das Programm läuft. Da die TSR-Programme (Hintergrundprogramme) beliebter werden, und der Platzbedarf des Betriebssystems wächst, wird es immer sinnvoller, daß die Programme nur so viel Speicher belegen, wie sie benötigen. Der Speicher eines PCs ist zu einer gemeinsam benutzten Ressource geworden. Welche Möglichkeit hat nun ein Programmierer, den Speicher für die Daten zu reservieren?

Speicherorganisation

Der Speicher eines Programms ist in Bereiche unterteilt, die *Heap*, *Stack*, *Daten* und *Code* genannt werden. Der Heap liegt am oberen Ende des Speicherbereichs eines Programms und ist die Quelle für dynamisch allokierten Speicher. Die Größe des Heaps hängt von der Größe der anderen Bereiche ab, da seine Größe dem verfügbaren Speicher, der nicht von anderen Bereichen belegt wird, entspricht.

Der Stack liegt unterhalb des Heaps im Speicher. Seine Größe stellt der Linker ein, entweder voreingestellt 2048 Bytes, oder die Größe, die dem Linker mit der Option `/STACK` angegeben wird. Der Datenbereich enthält statische und externe Variablen. Die Programmbefehle sind im Codebereich angesiedelt.

Allokieren von Speicher

Verfügt ein Programm über Speicher für seine ausschließliche Verwendung, sagt man, der Speicher ist von diesem Programm allokiert. C-Programme bieten zwei Möglich-

```
/*****************************************************************
 * SUM_NUM.C: Eingabe und Aufsummierung einer variablen Anzahl
 * von Integern. Zeigt Liste der Zahlen und die Aufsummierung.
 */

main()
{
   long  *zahlen;            /* Zeiger auf das Zahlenarray */
   short anzahl;             /* Anzahl der einzugebenden Zahlen */
   short ind_zahl;           /* Index in das Zahlenarray */
   long  summe;              /* Summe der Zahlen des Array's */

   printf("Wie viele Zahlen summieren? ");
   if(1 != scanf("%hd", &anzahl) || anzahl < 2)
      exit(1);               /* Programmende */

   /* Dynamisches Allokieren von anzahl long's */
   zahlen = (long *) calloc(anzahl, sizeof(long));

   for(ind_zahl = 0; ind_zahl < anzahl; ++ind_zahl) {
                             /* Zahleneingabe */
      printf("Eingabe #%d: ", ind_zahl + 1);
      if(1 != scanf("%ld", &zahlen[ind_zahl]))
         anzahl = ind_zahl;           /* Abbruch */
      }
   for (summe = ind_zahl = 0; ind_zahl < anzahl; ++ind_zahl)  {
      summe += zahlen[ind_zahl];  /* Zahl zur Summe addieren */
      printf("%3d:   %10ld %10ld\n",     /* Ergebnis anzeigen */
           ind_zahl + 1, zahlen[ind_zahl], summe);
      }
   printf("\nDie Summe der %d eingegebenen Zahlen ist %ld\n",
        anzahl, summe);
}
```

Listing 2: *Das Programm* `SUM_NUM.C`.

```
main()
{
   long  long_array[100]; /* Gewöhnliches long-Array */
   long  *p_longs;        /* Zeiger auf den Datentyp long */

   p_longs = &long_array[0]; /* Zeiger auf den Beginn des */
                             /* Arrays initialisieren */
   p_longs[8] = 32;
   printf("%ld gleich %ld\n", p_longs[8], long_array[8]);
   printf("sizeof p_longs %d, sizeof long_array %d\n",
          sizeof p_longs, sizeof long_array);
}
```

Die Ausgabe:

```
32 gleich 32
sizeof p_longs 2, sizeof long_array 400
```

Listing 3: *Die Verwendung von Zeigern mit Arrays.*

keiten der Speicherallokierung. Der gebräuchlichste Weg ist die einfache Deklaration von Variablen. Der andere Weg ist das dynamische Allokieren durch den Aufruf von Standardfunktionen, wie `malloc`, `calloc` und `realloc`.

Der Speicher für statische und externe Variablen wird nur einmal vor der Programmausführung allokiert und initialisiert, nämlich beim Laden von der Diskette in den Arbeitsspeicher. Er bleibt allokiert, bis das Programm endet.

Alle anderen Variablen, die mit der Speicherklasse `auto` und `register`, werden bei jedem Eintritt in die Funktion (Block), in der sie deklariert sind, allokiert. Der Speicher für `auto`- und `register`-Variablen wird beim Verlassen der Funktion (Block) wieder freigegeben (deallokiert).

Da der Speicher für die statischen und die externen Variablen während der gesamten Laufzeit benötigt wird, spart das Vermindern dieser Variablen RAM. `Auto`- und `register`-Variablen hingegen werden nur allokiert, während die Funktion aktiv ist (auf dem Stack). Eine Funktion ist aktiv, wenn sie von `main` oder einer anderen Funktion aufgerufen wird. Lebensdauer, Gültigkeit und Initialisierung sind andere Faktoren, wenn Sie Speicherklassen für Ihre Variablen auswählen.

Zum dynamischen Allokieren muß die Größe des benötigten Speichers bekannt sein. Das C-Schlüsselwort `sizeof` wird verwendet, um die Anzahl der Bytes der verwendeten Variablen zu finden. Der Operator `sizeof` muß auch für in Klammern eingeschlossene Datentypen verwendet werden. Das bedeutet, daß das Programm den Wert des Ausdrucks `sizeof(int)` mit zwei oder vier berechnen kann, abhängig von der Übersetzung auf einer 16- oder 32-Bit-CPU (*Listing 1*).

Arrays variabler Länge

Die dynamische Allokierung eines Speicherblocks findet zur Laufzeit des Programms statt. Die Größe wird durch Funktionsargumente beim Aufruf der Standardfunktionen, wie `malloc`, `calloc` oder `realloc`, festgelegt. Dies ist unterschiedlich zur Deklaration eines Arrays, die an eine Konstante oder an einen Konstantenausdruck gebunden ist.

Das Grundprogramm `SUM_NUM` in *Listing 2* verwendet die Standardfunktion `calloc`. In `SUM_NUM` gibt `calloc` die Startadresse des Speicherblocks zurück, dessen Größe durch die Übergabe der Funktionsargumente *Anzahl* und *Größe der Elemente* festgelegt ist. Sie brauchen nur einen Zeiger auf den Datentyp, der gespeichert werden soll deklarieren und das Funktionsergebnis diesem zuweisen wie im folgenden gezeigt:

```
long     *zahlen
          .
          .
zahlen = (long *) calloc(anzahl, sizeof(long));
```

Diese zwei Statements erzeugen ein dynamisch allokiertes Array von `anzahl` `long`-Integerzahlen, und initialisieren den Zeiger `zahlen` auf das Array.

Das `(long *)` vor `calloc`, welches ein `cast`-Operator ist, drückt Zeiger auf den Datentyp `long` aus, und ist optional. Es ist eine Typumwandlung, die Compilerwarnungen wie »different levels of indirection« vermeidet.

```
MEMO     TXT      46  11-18-86   6:26p
DIROUT   TXT    1206   1-17-87   7:25p
```

Listing 4: *Beispieldaten von einem DIR-Befehl.*

Wird eine Funktion nicht vor der Verwendung deklariert, oder wird ihr Rückkehrtyp nicht explizit (wie zum Beispiel `long`) deklariert, nimmt der Compiler an, daß die Funktion ein `int`-Ergebnis liefert. Die Zuweisung des Funktionsergebnisses von `calloc` ohne `(long *)` an `zahlen` hätte als Ergebnis die Zuweisung einer `int`-Zahl an eine Variable, die als Zeiger auf `long` deklariert ist. Dies veranlaßt den Compiler eine Warnung auszugeben.

Ist der angeforderte Speicher nicht mehr verfügbar, liefert `calloc` den Wert `NULL`, der in `stdio.h` als `0` definiert ist. Prüfen Sie etwa den Funktionsaufruf nicht auf den speziellen Rückgabewert 0, und speichern Daten an dieser Adresse, erhalten Sie am Programmende die Fehlermeldung »null pointer assignment«. Das Schreiben von Daten in nicht allokierten Speicher kann leicht unbemerkt bleiben und schwer zu findende Fehler verursachen. Schwer zu finden deshalb, weil das Programm nur in einigen Fällen versagen kann, die bei der Programmerstellung eventuell nicht berücksichtigt wurden.

Zeiger und Arrays

Das Verständnis von Zeigern und Arrays ist sehr wichtig in C, speziell bei der Speicherverwaltung. Zeiger- und Array-Datentypen sind sehr ähnlich - vielleicht mehr, als Sie glauben. Immer wenn Sie in einem Ausdruck ein Array verwenden, etwa bei der Parameterübergabe zu einer Funktion oder beim Erhalten des Wertes eines Elements, wird die Startadresse des Arrays verwendet. Wenn Sie eine Zeigervariable verwenden, die auf ein Array desselben Datentyps zeigt, so können Sie den Zeiger in Ausdrücken verwenden, als wäre er das Array *(Listing 3)*.

Die Tatsache, daß sich Zeiger ähnlich wie Arrays verhalten, bedeutet, daß Sie Elemente des Arrays mit der Variablen `zahlen` bearbeiten können *(Listing 2)*, so als ob `zahlen` ein Array von `long`-Integern wäre. Die Adresse eines Elements, auf die `zahlen` im Ausdruck `&zahlen [anzahl]` zeigt wird der Funktion `scanf` übergeben. Der Wert des Elements wird zur Summe im Statement `summe += zahlen[anzahl];` addiert.

Selbst wenn Sie ein C-Neuling sind, werden Sie damit jedesmal konfrontiert, wenn Sie eine Funktion programmieren, die mit Arrayelementen arbeitet, die als Parameter übergeben werden. Die Parameter einer Funktion, die Daten eines Arrays erhalten, werden mit leeren eckigen Klammern deklariert, wie im Ausdruck `get_param[]`. C übergibt an diese Funktion die Startadresse des Arrays.

```
#include <stdio.h>                  /* Für NULL-Zeiger Definition */
#define  DIR_LINE_LEN  39           /* Zeilenlänge von DIR */
struct s_dir_node {
      struct s_dir_node *next; /* Zeiger auf nächsten Knoten */
                               /* oder NULL */
      char   dir_line[DIR_LINE_LEN + 1];  /* DIR Ausgabezeile */
      };

struct    s_dir_node   *p_head; /* Zeiger zum Listenbeginn */
struct    s_dir_node   *p_new;  /* Zeiger zu neuem Eintrag */
p_head = NULL;                  /* Liste ist leer */
```

p_head	0000
p_new	????

Listing 5: *Deklaration der Kennung der Verbindungsstruktur.*

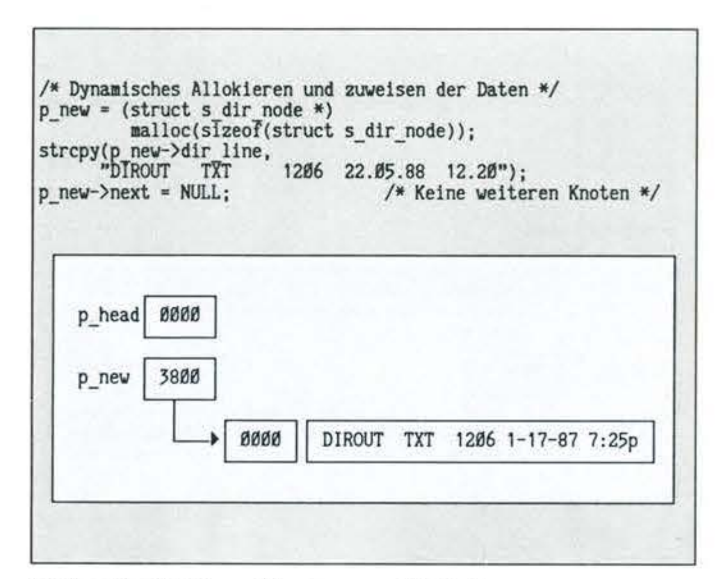
Die Funktion behandelt den Parameter als Array, obwohl der Parameter eigentlich ein Zeiger ist. Der einzige Unterschied zwischen Zeigern und Arrays besteht bei der Verwendung des `sizeof`-Operators. Im *Listing 3* gibt der `sizeof`-Operator unterschiedliche Werte für `p_long` und `long_array` zurück.

Die Konsequenz dieser Art der Paramterübergabe ist, daß die aufgerufene Funktion nicht die Dimension eines an sie übergebenen Arrays kennt. Da das Array für sie nicht verfügbar ist, kann sie die Größe nicht testen. Stattdessen ist der übergebene Wert ein Zeiger auf das erste Element, und eine Zeigervariable ist 2 oder 4 Bytes lang (`int` oder `long` auf einer 16-Bit-Maschine):

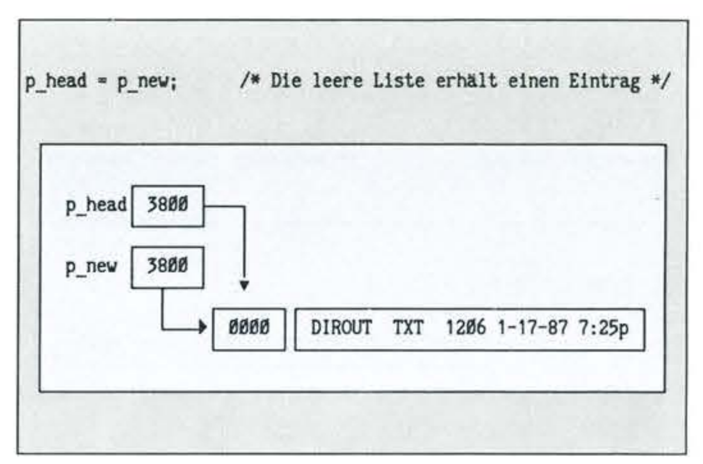
Rückgabe des Speichers

Beim Programmende wird der ganze allokierte Speicher zurückgegeben. Dies beinhaltet auch den von Funktionen wie `calloc` allokierten dynamischen Speicher. Wird allokierter Speicher deallokiert, so steht er für weitere Speicheranforderungen zur Verfügung. Benötigt Ihr Programm mehrere Allokierungen, so können Sie die Funktion `free` aufrufen, um nicht zu wenig Speicher zur Verfügung zu haben. Rufen Sie `free` auf, um nicht mehr benötigte Speicherblöcke zurückzugeben. Die Funktion `free` benötigt als Argument dieselbe Adresse, die ein früherer Funktionsaufruf von `calloc`, `malloc` oder `realloc` als Ergebnis lieferte, wie zum Beispiel:

```
zahlen = (long *)calloc(anzahl, sizeof(long));
free(zahlen);
/* Deallokiert Speicher, auf den zahlen zeigt */
```



Listing 6: *Allokieren für eine neue Verkettung.*



Listing 7: *Die erste Verkettung.*

Übergeben Sie `free` nie eine Adresse, die Sie nicht vorher von `calloc`, `malloc` oder `realloc` erhalten haben, sonst gibt es Probleme. Jedem Block stehen Informationen zur internen Systemverwaltung voran, wie die Größe des Blocks und ein Zeiger auf den nächsten Block. Ist diese Verwaltungsinformation zerstört, so ist der Pool des freien Speichers »verschmutzt«. Die dynamischen Allokierfunktionen tun so, als ob freier Speicher bereits allokiert wäre, oder schlimmer, daß bereits allokierter Speicher frei für die Wiederallokierung ist.

Um die Konzepte für die dynamische Speicherverwaltung zu verdeutlichen, sehen wir uns eine Applikation an, die ein Inhaltsverzeichnis sortiert. Das Inhaltsverzeichnis einer Diskette enthält eine unbekannte, nicht vorhersagbare Anzahl von Einträgen. Ein Inhaltsverzeichnis-Sortierprogramm kann daher zwei verschiedene Wege beschreiten. Entweder wird eine festgelegte Größe vorher bestimmt, oder die Sortierroutine kann so ausgelegt werden, daß sie allen verfügbaren Speicher benützt. Obwohl das Programm einfach erscheint, ist es ein gutes Beispiel für die dynamische Speicherallokierung, und zeigt die Speicherverwaltungstechnik der verketteten Listen.

Definition des Problems

Die Anzahl der Dateien in einem Inhaltsverzeichnis kann sich unvorhersagbar ändern. Inhaltsverzeichnisse können von zwei (».« und »..« - ein leeres Unterverzeichnis) bis zu der jeweiligen Obergrenze der Betriebssystemversion möglichen Einträge aufweisen. Da sich die Obergrenze mit der nächsten Version ändern kann, sollte ein Dienstprogramm zum Sortieren eines Inhaltsverzeichnisses diese Änderungen ohne Neuprogrammieren akzeptieren.

Das Kommando DIR gibt die Informationen über die Dateien in einem Inhaltsverzeichnis aus. Die Reihenfolge der Anzeige ist aber nicht sonderlich hilfreich. Eine Lösung für beide Probleme wird im folgenden erörtert.

Das MS-DOS Filterprogramm `SORT` kann zum Sortieren der Anzeige verwendet werden. Sie müssen `SORT` nur mitteilen, ab welcher Spalte es sortieren soll und es erledigt die Aufgabe. Die Verwendung einer Pipe mit den Kommandos `SORT` und `DIR` ist nützlich aber etwas eingeschränkt.

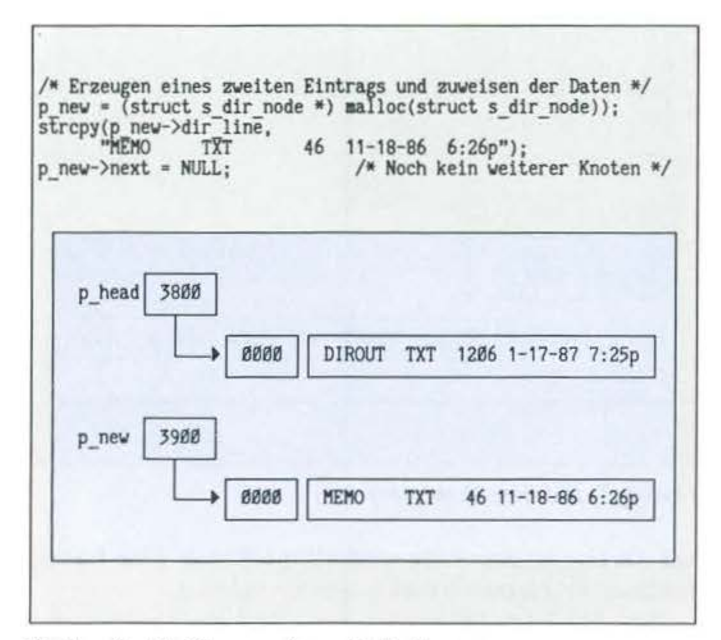
Des weiteren handhabt `SORT` einige Dinge recht kläglich. Eine am frühen Nachmittag geänderte Datei wird vor einer am späten Vormittag geänderten Datei angezeigt, da MS-DOS (US) »a« und »p« zum Repräsentieren von A.M. (vormittags) und P.M. (nachmittags) statt der 24-Stunden-Anzeige verwendet. Die Kombination von `DIR` und `SORT` kann hilfreich sein, aber es wird etwas Besseres benötigt.

Verkettete Listen

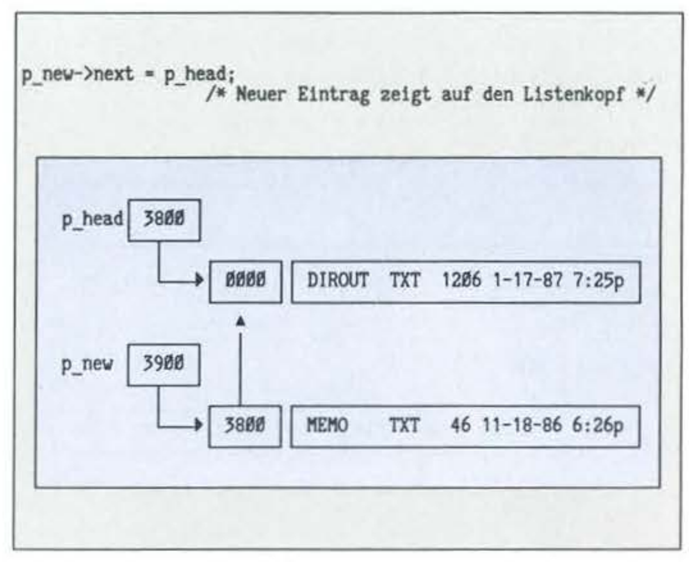
Verkettete Listen sind mächtige und flexible Werkzeuge um dynamisch allokierten Speicher zu verwalten. Sie eignen sich in Anwendungen, wo Daten unterschiedlicher Typen (Integer, Zeichen und Fließkommazahlen) in Strukturen mit nicht vorhersagbarer Länge verkettet werden können. Jedes Element einer verketteten Liste ist wie ein Satz in einer speicherresidenten Datendatei. Ein Satz enthält Schlüssel für andere Sätze, indem er darauf zeigt. Diese Sätze sind in Strukturen von einer einfachen Einzelverkettung bis zu exotischen Verkettungs-Netzwerken verkettet.

Verkettete Listen existieren in unterschiedlichen Größen und Gestalten, mit einigen Gemeinsamkeiten und einer einheitlichen Terminologie. Sie können in der Struktur ziemlich komplex sein. Wir sehen uns die einfachste Form, die einfach verkettete Liste an, die eine Reihe von Elementen, Knoten genannt, enthält, die zu einer Kette verknüpft sind. Es ist nur möglich, sich durch diese Liste in einer Richtung zu bewegen, vom Startknoten der Liste, dem Kopf, zu dem Endeknoten, dem Schwanz.

Durch die Einfachheit, mit der Knoten zwischen zwei Knoten eingefügt werden können, eignen sich Listen vor-



Listing 8: *Allokieren weiterer Verkettungen.*



Listing 9: *Die zweite Verkettung.*

züglich zum Speichern von Daten in unterschiedlichen Sortierungen. Wird jeder neue Knoten einer Liste am Kopf eingefügt, dann verhält sich die Liste wie ein Push-Down-Stack (auch *Last in First out* oder LIFO genannt). Werden neue Knoten nur an das Ende der Liste angehängt, ist das Ergebnis eine *First in First out*-Schlange (FIFO). Jede Ordnung ist möglich, da neue Knoten überall eingefügt werden können. Das alphabetische Sortieren von Text ist leicht zu bewerkstelligen.

Die Daten im *Listing 4* werden vom DIR-Kommando ausgegeben, und werden benutzt, um zwei Arten von verketteten Listen zu erzeugen: zuerst einen Push-down-Stack, und dann eine sortierte Liste. Das Erzeugen eines Push-down-Stacks von einer leeren Liste ist in den *Listings 5, 6, 7, 8, 9* und *10* dargestellt. Die Liste selbst ist im *Listing 11* dargestellt. Ein Speicherabbild, das den Zustand der Liste nach jedem Codefragment zeigt, ist den *Listings 5, 6, 7, 8, 9* und *10* beigefügt.

Die Struktur `s_dir_node` ist im *Listing 5* zusammen mit den Zeigern `p_head` und `p_new`, die auf diese Strukturen zeigen, deklariert. Das Strukturmitglied `next` ist ebenfalls ein Zeiger auf eine `s_dir_node`-Struktur. Diese Art der Struktur wird *selbstreferenzierend* genannt, da sie einen Zeiger auf den eigenen Datentyp enthält. Das Mitglied `next` dient zum Verketten der Strukturen, um die Liste zu bilden. Dem Zeiger auf den Kopf, `p_head` wird der Wert `NULL` (aus `stdio.h`) zugewiesen, um die leere Liste zu erzeugen.

Der Speicherplatz zum Unterbringen eines einzelnen `s_dir_node`-Knotens wird dynamisch mit der Funktion `malloc` in *Listing 6* allokiert. Das Argument, das `malloc` übergeben werden muß, ist die Anzahl der Bytes, die benötigt werden. Der Zeiger, der von `malloc` zurückgegeben wird, ist die Adresse des neu allokierten Blocks, aber vom Typ Zeiger auf `char`, und muß deshalb auf einen Zeiger vom Typ `s_dir_node` mit dem cast `(struct s_dir_node *)` konvertiert werden. Die typkonvertierte Adresse wird der Variablen `p_new` als letzte Aktion des ersten Statements zugewiesen. Das Speicherdiagramm zeigt, daß der neue Knoten an Adresse 3800 steht.

Die letzten beiden Statements im *Listing 6* weisen dem allokierten Knoten Werte mit dem Operator `->` zu. Der Ausdruck `p_new->dir_line` ist das Mitglied `dir_line` in der Struktur `s_dir_node`, auf die `p_new` zeigt. `p_next->next` wird `NULL` zugewiesen, um es als Schwanz der Liste kenntlich zu machen. Dieser Knoten ist zugleich Kopf der Liste.

Durch die Zuweisung in *Listing 7* zeigt `p_head` auf denselben Knoten wie `p_new`. Im *Listing 9* wird ein neuer Knoten an Adresse 3900 allokiert. Die kombinierten Aktionen in den Listings 9 und 10 plazieren in Push-down-Stack-Weise den zweiten Knoten an den Kopf der Liste vor den vorherigen Kopf.

Die `for`-Schleife im *Listing 11* zeigt jeden Knoten an, beginnend beim Kopf zum Ende fortschreitend, einen je Durchlauf. Ein interessanter Aspekt der Schleife ist, daß statt eines Hoch- oder Niederzählens durch eine Liste von Zeigern geschritten wird. Der Ausdruck `p_new = p_new->next` zeigt dieses Durchschreiten, indem `p_new` der Wert des Mitglieds `next` zugewiesen wird.

Beachten Sie, daß im Formatstring für `printf`, innerhalb der Schleife, der Formatbezeichner `%u` (unsigned integer) zur Anzeige der Adressen verwendet wird, da Adressen nie negativ sind.

Da Sie jetzt erste Erfahrungen im Erstellen von verketteten Listen haben, schauen Sie sich einen Augenblick den Vergleich zwischen verketteten Listen und Arrays im *Listing 12* an. Verkettete Listen vereinfachen den Umgang mit

CHIP WISSEN



Rudolf Wolff:

Professionell arbeiten mit Ventura Publisher

264 Seiten, zahlr. Bilder, Hardcover
43,– DM/ISBN 3-8023-**0213**-3

Professionell arbeiten mit Ventura Publisher ist sowohl eine Einführung in die Programmvielfalt des Xerox Ventura Publisher, als auch und gerade ein Praxishandbuch, das dem Leser immer dann hilfreich zur Seite steht, wenn Gestaltungs- und Typografieprobleme auftreten. Nach der Einleitung über Voraussetzungen und Möglichkeiten des Ventura-Publisher und einigen Ausführungen zu Satz-, Layout- und Typografiegrundsätzen folgt die ausführliche Einweisung in das Programm, um die angebotenen Möglichkeiten sicher und rationell zu nutzen. Der Praxisteil soll mit Übungen und Beispielen die bis dahin erworbenen theoretischen Kenntnisse vertiefen und dem Anwender Sicherheit vermitteln.

Hans-Peter Förster/Martin Zwernemann:

Word 4.0 kurz und bündig

256 Seiten, 67 Bilder, Hardcover
43,– DM/ISBN 3-8023-**0215**-X

Dieses Arbeitsbuch richtet sich an interessierte Einsteiger sowie an Word-4.0-Anwender, die schnell den Umgang mit dem komfortablen Textverarbeitungsprogramm erlernen wollen. Es ist so aufbereitet, daß es auch als Begleitmaterial für Word-4.0-Schulungen geeignet ist. Aber auch zum Selbststudium ist es in Ergänzung zum Word-4.0-Handbuch ein Hilfsmittel, das viele Tips und Tricks verrät.

* Wichtige Grundfunktionen
* Texte erstellen, korrigieren, speichern
* Texte laden, korrigieren, gestalten
* Texte direkt/indirekt formatieren
* Ausdruck
* Textbausteine
* Rechtschreibprogramm u.a.

Haben Sie schon den neuen „CHIP- Katalog 1988"? Bestellen Sie gleich!



VOGEL Buchverlag Würzburg
Postfach 6740 · 8700 Würzburg 1

Starkes Werkzeug für komplexe Applikationen : **Das Datenbank-Entwicklungssystem für C-Programmierer**

UNIX, VAX, MS-DOS, OS/2

db_VISTA III - zwei Vorteile in einem: relationale Datenbanken sind flexibel, Netzwerkdatenbanken dagegen extrem schnell und frei von Datenredundanzen. db_VISTA verknüpft beide Vorteile: extrem schneller Datenzugriff bei geringem Speicherbedarf und flexibler Struktur. db_VISTA: komplett in C (Source-Code erhältlich). SQL-Schnittstelle. Portabel und offen. Schnittstelle zu Lotus 1-2-3 u.a. durch WKS-Library. Umstrukturierung mit db-REVISE. Keine Run-Time-Gebühren!

Die db_VISTA-Vorteile

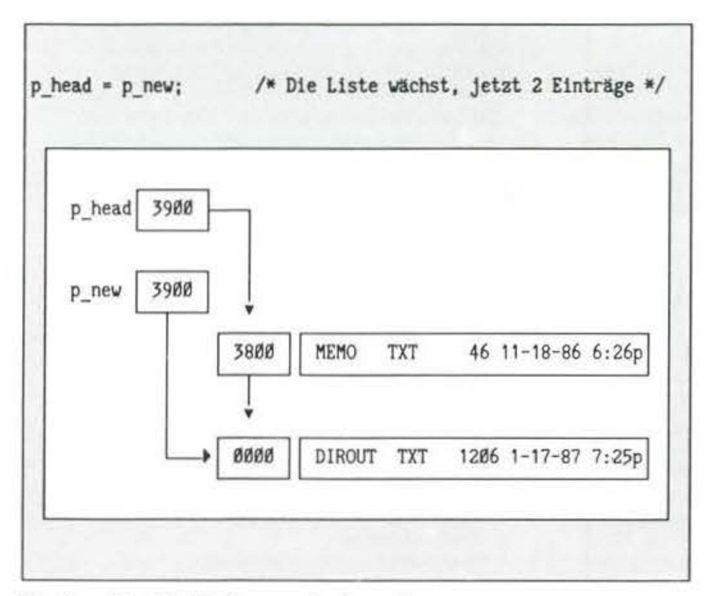
- Mehrplatzunterstützung für Netzwerke oder Multiuser-Systeme (z.B. UNIX)
- Multiuser-Versionen für UNIX, VAX VMS, MS-DOS, OS/2
- Transaction Processing (Records werden im Zusammenhang aktualisiert)
- Hilfsprogramme (Import/Export ASCII, Überprüfung der Datenintegrität u.a.m.)
- Automatische Daten-Wiederherstellung nach Datenverlust
- Umstrukturierungsprogramm db_REVICE
- Gleichzeitige Bearbeitung mehrerer unterschiedlicher Datenbanken
- Referenzhandbuch, User's-Guide
- db_Vista-**Schulungen**
- 30 Tage Rückgaberecht für Neukunden
- 60 Tage kostenloser Support
- Support-Vertrag
- Hotline-Service
- **keine Runtime-Lizenz-Gebühren!**





Testen Sie db_VISTA: 30 Tage Rückgaberecht , 60 Tage kostenloser Support

ESM GmbH · Stuttgarter Straße 90 · 7447 Aichtal-Aich · Tel.: 07127/52 44



Listing 10: *Die Verkettung ist beendet.*

nicht vorhersagbaren Datenmengen. Die Folge der Daten kann leicht geändert werden, da leicht ein Knoten oder eine Folge von Knoten zwischen zwei existierende Knoten der Liste eingesetzt werden können. Ebenso einfach ist das Herausnehmen von Knoten oder einer Folge von Knoten.

Der Zugriff auf unterschiedliche Elemente eines Arrays ist sehr schnell und erlaubt wahlfreien Zugriff. Auf die Elemente einer verketteten Liste muß in der Reihenfolge vom Kopf Knoten für Knoten bis zum Schwanz zugegriffen werden. Verschiedene Pfade (Ordnungen) durch eine verkettete Liste können durch Hinzufügen von neuen Zeigern in der Struktur erreicht werden. Die Bezeichnung doppelt verkettete Liste steht für eine Liste, die ebenso Zeiger zum Vorgängerknoten enthält. Der zusätzliche Aufwand des Speicherns der Verkettungszeiger in jedem Knoten ist ein Preis, der bei verketteten Listen, aber nicht bei Arrays zu zahlen ist. Diese Tatsachen in Erinnerung behaltend, wollen wir mit einer realeren Applikation für verkettete Listen fortfahren.

Das Programm BY_TIME

Das Programm BY_TIME kann zum Sortieren der Ausgabe des DIR-Kommandos benutzt werden. Die Information wird in der Reihenfolge des Datums und der Uhrzeit angezeigt, wobei die jüngsten zuerst angezeigt werden. Das Sortieren wird mit Hilfe des Einfügens von Knoten in eine verkettete Liste durchgeführt, die ständig in der richtigen Sortierfolge vorliegt. Das DOS-Kommando zum Ausführen des Kommandos BY_TIME, um `*.EXE`-Dateien in Datum- und Uhrzeit-Reihenfolge auszugeben, sehen Sie in *Listing 13*, zusammen mit einer Beispielausgabe.

Die Organisation und Struktur des Programms BY_TIME zeigt einen Codierstil, der zum Entwickeln größerer Applikationen geeignet ist. Die `main`-Funktion

```
/* Nach dem Anlegen der verketteten Liste wird diese in einer
   for-Schleife angezeigt */

for(p_new = p_head; p_new != NULL; p_new = p_new->next)
   printf("Adresse: %4u, next: %4u, dir_line: %-.12s}n",
         p_new, p_new->next, p_new->dir_line);
```

Ausgabe:

```
Adresse: 3900, next: 3800, dir_line: MEMO     TXT
Adresse: 3800, next:    0, dir_line: DIROUT   TXT
```

Listing 11: *Ergebnis der Ausgabe der verketten Liste.*

verbirgt die Details der Datenmanipulation, die in den unteren Funktionen durchgeführt werden, gibt aber einen schnellen Überblick über die durchgeführte Bearbeitung. Jede Funktion ist kurz und prägnant und erledigt eine einzige Aufgabe. Keine externen Daten werden benötigt. Die Datenpfade, in denen die Informationen fließen, werden durch die Argumentübergabe klar.

Das Programm BY_TIME erhält man durch das getrennte Übersetzen aller C-Quelldateien und das Linken der entstandenen Objektdateien (der *Listings 14, 15, 16, 17, 18, 19* und *20*).

Jeder Knoten in der verketteten Liste von BY_TIME ist eine vertraute Struktur mit zwei Mitgliedern, einem Zeiger auf einen anderen Knoten und einem String von Informationen über die Dateien, die von DIR ausgegeben werden. Die Struktur der Knoten ist in der Headerdatei `BY_TIME.H` im *Listing 14* definiert und kann somit in jede Quelldatei, die sie benötigt, eingelesen werden.

Die erste Aktion der Funktion `main` im *Listing 15* ist der Funktionsaufruf von `bgn_list` mit dem Argument eines Kopfknoten, der als `s_dir_node`-Struktur deklariert ist. Die Funktion `bgn_list` im *Listing 16* weist Dummydaten zu, womit gewährleistet ist, daß der Knoten immer der Kopf der Liste bleibt. Dieser spezielle Kopf hilft, das Programm sauber zu gestalten, da der Kopfknoten beim Einfügen nicht als Spezialfall berücksichtigt werden muß. Der Kopf der Liste ist nicht dynamisch allokiert, sondern ist eine gewöhnliche `auto`-Strukturvariable.

Die initialisierte Liste ist jetzt vorbereitet, um mit der Information der Dateien zu wachsen. Die `while`-Schleife in `main` führt wechselweise `get_dir`- und `sav_dir`- Funktionsaufrufe durch, bis `get_dir 0` zurückgibt. Jeder Funktionsaufruf von `get_dir` (*Listing 17*) fügt eine Zeile von Dateiinformationen in den übergebenen Puffer. Er gibt bei Erfolg `1`, ansonsten `0` zurück (Ende der Eingabe).

Nach der Rückgabe der ersten `DIR`-Information von der Funktion `get_dir` wird der erste Block dynamischen Speichers allokiert und wie im *Listing 18* gezeigt zur Verwendung an die Funktion `sav_dir` übergeben.

Verkettete Listen	Arrays
Variable Länge	Feste Dimensionierung mit Maximalgröße
Kann zusätzlichen Speicher ansprechen	Unterstützt keinen zusätzlichen Speicher
Schnelles Einfügen und Löschen	Löschen und Einfügen langsam
Kein wahlfreier, nur sequentieller Zugriff	Direktzugriff
Zusätzlicher Speicher für Knotenzeiger	Kein zusätzlicher Speicherbedarf
Programme zur Listenmanipulation sind schwer zu lesen	Programme zur Arraybearbeitung sind leicht lesbar

Listing 12: *Vergleich von verketteten Listen und Arrays*

Die Funktion `sav_dir` hat die Aufgabe, die neuen Knoten so in die Liste einzufügen, daß sich die Liste in der entsprechenden Reihenfolge befindet.

Das Einfügen beginnt mit einer Schleife, die die Liste nach dem korrekten Einfügepunkt durchsucht. Die Funktion `time_b4` im *Listing 19* wird in der `for`-Schleife von `sav_dir` als Bedingungstest aufgerufen. Die Funktion `time_b4` vergleicht Datum und Uhrzeit des Eingabepuffers mit dem Datum und der Uhrzeit im Knoten der Liste und gibt 1 zurück, wenn die Pufferzeit älter ist, ansonsten 0.

Speicher für neue Knoten wird in der Funktion `sav_dir` durch den Funktionsaufruf von `malloc` dynamisch angefordert. Beachten Sie, daß ein Test auf `NULL` durchgeführt wird, um sicherzustellen, daß genügend Speicher für den neuen Knoten vorhanden ist. Die letzten beiden Statements in `sav_dir` lösen die Verbindung zwischen den beiden Knoten, fügen den neuen Knoten ein, und erzeugen dadurch eine geordnete Liste, die um ein Element länger ist.

Jede `DIR`-Ausgabezeile wird von der Funktion `sho_list` im *Listing 20* angezeigt, indem sie vom Kopf zum Schwanz die Liste durchwandert. Dies wird durchgeführt, nachdem die Eingabe beendet und die Liste erstellt ist. Das Programm in `sho_list` geht durch die Liste, den Datenteil jedes Knotens anzeigend, so wie es auch die Schleife in *Listing 11* erledigte.

Wilde Zeiger

Wie schon am Beispiel von `free` gezeigt, muß bei der Speicherverwaltung in C aufgepaßt werden, ganz besonders beim Datenzugriff über Zeiger. Das Programm `DANGLE.C`

```
DIR *.EXE | BY_TIME

BY_TIME  EXE    5654   1-26-87   8:44p
VWC      EXE   16886   1-17-87   8:22p
GREP     EXE    9336   4-06-86   6:34a
PR       EXE   11604   3-16-86  11:07a
CFIND    EXE    8146   6-20-85   5:30p
CHARCNT  EXE    7316   4-07-85   9:63p
LISTING  EXE    6226   4-07-85   9:27p
VISIBLE  EXE    6402   4-07-85   9:24p
```

Listing 13: *Ausgabe des Programms* `BY_TIME`.

im *Listing 21* ist ein Beispiel für Speicher-Mismanagement, da es Speicher im nicht allokierten Speicher anspricht. Die Resultate ergeben beim ersten Anblick keinen Sinn, bis Sie erkennen, daß ein häßlicher Fehler im Spiel ist. Die Ausgabe besteht aus drei verschiedenen Zeilen von Schrott. Versuchen Sie es selbst.

Der Speicher, auf den `p_next` zeigt, war in keinem der drei Fälle allokiert, in denen die Adresse an die Funktionen `puts` und `printf` zur Ausgabe übergeben wurde. Der Speicher war allokiert, während `str_dang` ausgeführt wird, aber nach der Rückkehr ist die Adresse der `auto`-Variablen `auto_string`, der Speicher für `auto_string` deallokiert. Automatische Variablen innerhalb einer Funktion werden deallokiert, wenn die Funktion zum Aufrufer zurückkehrt. Danach wird der Speicher von `auto_string` reallokiert, so daß der String, auf den `p_next` zeigte, sich ändert, wie wenn böse Geister im Spiel wären. Der nächste Gebrauch war für `printf` oder eine seiner Unterfunktionen gedacht.

Speicher für alle statischen oder externen Variablen eines Programms bleibt für den gesamten Programmlauf allokiert. Wäre die Deklaration von `auto_string` durch das Wort `static` eingeleitet worden, erhielten Sie statt der drei falschen Zeilen die drei gleichen Zeilen 123456789.

Zusammenfassung und Ausblick

Programme, die unvorhersehbare Mengen von Daten im Arbeitsspeicher bearbeiten müssen, profitieren von dynamisch allokierten Arrays. Diese Arrays können aus einfachen Datenelementen oder aus komplexen Strukturen bestehen. Nach dem Allokieren kann auf diese dynamischen Arrays durch die Verwendung der eckigen Klammern `[]` zugegriffen werden, um einzelne Elemente anzusprechen, so wie bei gewöhnlichen Arrays.

Bedarf die Datenbearbeitung eine Änderung in der Reihenfolge der Elemente, dann sollten verkettete Listen in Erwägung gezogen werden. Die Entscheidung wird anhand des Zugriffs auf die Daten getroffen. Wird häufig wahlfreier Zugriff benötigt, oder wird die Liste nur in sequentieller Folge bearbeitet? Ist der Mehraufwand von Knotenzeigern in jedem Knoten vertretbar? Sind die Personen, die die

```
/*****************************************************************
 * BY_TIME.H: Header-Datei für das Programm BY_TIME, die die
 *            verkettete Listenstruktur definiert
 */
#define  DIR_LINE_LEN   39           /* Zeilenlänge von DIR */
struct s_dir_node {    /* Knotenstruktur der verketten Liste */
      struct s_dir_node *next; /* Zeiger zum nächsten Knoten */
                               /* oder NULL */
      char dir_line[DIR_LINE_LEN + 1];  /* DIR Ausgabezeile */
      };
```

Listing 14: BY_TIME.H.

```
/*****************************************************************
 * BY_TIME.C: Filter, der die DIR Ausgabe in einer verketteten
 *            Liste sortiert. Hier die Hauptfunktion des
 *            Programms.
 */

#include <stdio.h>          /* Für BUFSIZ */
#include "by_time.h"        /* Struktur der verketteten Liste */

main()
{
   struct s_dir_node head;/* Listenkopf */
   char   in_buf[BUFSIZ]; /* Eingabepuffer für DIR Ausgabe */

   bgn_list(&head);            /* Liste initialisieren */
   while (get_dir(in_buf))    /* DIR-Info für nächste Datei */
      sav_dir(in_buf, &head);/* Speichern in einer */
                              /* sortierten verketteten Liste */
   sho_list(&head);            /* Anzeige der Liste */
}
```

Listing 15: BY_TIME.C.

Programme warten, sattelfeste C-Programmierer, die sich mit Zeigern, dem dynamischen Allokieren und der Theorie der verketteten Listen auskennen?

Hier wurde nur die einfach verkettete Liste besprochen. Die doppelt verkettete Liste, die vor- und rückwärts durchlaufen werden kann, wurde nur erwähnt. Ein Baum ist ein weit komplexerer Typ einer verketteten Liste. Der Kopf eines Baums kann zwei oder mehrere Zeiger auf Strukturen haben, die oftmals Nachfolger genannt werden, wie in einem Familienstammbaum. Jeder der Nachfolgerknoten kann wieder mehrere Nachfolger haben. Knoten ohne Nachfolger werden Blätter genannt. In einem Baum sind keine Schleifen erlaubt, ebenso wie eine Person nicht sein eigener Vater oder Opa sein kann. Anwendungen für Bäume sind Sortieren, Ausdrucksbewertung in Compilern und Übersetzern sowie das Bearbeiten von Verzeichnissen und Unterverzeichnissen auf Datenträgern.

Für mehr Informationen über das Speichermanagement, die dynamische Allokierung und verkettete Listen empfehle ich das Kapitel »Optimale Speicherausnutzung« in meinem Buch »Variationen in C« (Vieweg/Microsoft Press, 1987).

Die Güte der Speicherverwaltung macht für den Anwender einen wirklichen Unterschied in der Güte eines Programms aus. Machen Sie Ihre Zeiger nicht wild, und Ihre Listen bleiben in Verbindung. Steve Schustak

```
/*****************************************************************
 * BGN_LIST.C: Initialisieren des Listenkopfs mit dem
 *             größtmöglichen Wert.
 */

#include <stdio.h>          /* Für NULL Zeiger */
#include "by_time.h"        /* Struktur der verketteten Liste */

bgn_list(p_head)
struct s_dir_node *p_head;/* Zeiger auf Listenkopf */
{
               /* Datum im Kopf ist größer als alle anderen */
   strcpy(p_head->dir_line,
         "ZZZZZZZZ ZZZ     99999  99-99-99  99:99p");
   p_head->next = NULL;      /* Kein Nachfolger - leere Liste */
}
```

Listing 16: BGN_LIST.C.

```
/*****************************************************************
 * GET_DIR.C: Eingabe der DIR-Information für die nächste Datei
 *            von stdin
 */

#include <stdio.h>          /* Für NULL Zeiger */

int   get_dir(buf)
char  *buf;                 /* Puffer zum Lesen und zur */
                            /* Rückgabe der Zeile */
{
      char  *rtn;           /* Rückgabe von gets() */


      /* Schleife:
         Eingabe bis Ende, oder bis Zeile
         mit Großbuchstaben beginnt */


      while ((rtn = gets(buf)) && /* Eingabe einer Zeile */
             (buf[0] < 'A' || buf[0] > 'Z'))
         ;
      return(rtn != NULL);
}
```

Listing 17: GET_DIR.C.

```
/*****************************************************************
 * SAV_DIR.C: Allokieren eines neuen Knotens und das
 *            Abspeichern der DIR-Information
 */

#include <stdio.h>          /* Für NULL Zeiger */
#include "by_time.h"        /* Struktur der verketteten Liste */

sav_dir(buf, p_head)
char *buf;                          /* Zeile der DIR Ausgabe */
struct s_dir_node *p_head;          /* Zeiger auf Listenanfang */
{
   struct s_dir_node *p_next;       /* Zeiger auf nächsten */
                                    /* Knoten in der Liste */
   struct s_dir_node *old_p_next;  /* Zeiger zum Nachfolger */
                                    /* des Vorgängers */
   struct s_dir_node *p_new;        /* Zeiger auf neuen Knoten */


   /* Schleife:
      Für jeden Knoten in der Liste bis zum Ende, oder
      bis der Punkt zum sortierten Einfügen erreicht ist */
```

```
    for (p_next = p_head->next, old_p_next = p_head;
            p_next && time_b4(buf, p_next);
            old_p_next = p_next, p_next = p_next->next)
      ;


    /* Dynamisches Allokieren neuer Knoten.
       Beachten Sie den cast (struct s_dir_node *)
       um eine Warnung zu vermeiden.
     */


    p_new = (struct s_dir_node *)
            malloc(sizeof(struct s_dir_node));

    if (p_new == NULL) {    /* malloc() Fehler, */
                            /* kein Speicher mehr */
       puts("Kein freier Speicher!!!");
       return;
    }

    strcpy(p_new->dir_line, buf),  /* Speichern der DIR Zeile im
                                      neu allokierten Speicher */
    p_new->next = old_p_next->next;/* Neuer Knoten zeigt auf
                                      Listenrest */
    old_p_next->next = p_new;      /* Einfügen des neuen
                                      Knoten in der Liste */
}
```

Listing 18: `SAV_DIR.C.`

```
/*****************************************************************
 * TIME_B4.C: Rückgabe von 1, falls Datum und Uhrzeit in buf
 *            früher ist, als von Knoten p_next, sonst 0.
 */

#include "by_time.h"       /* Struktur der verketteten Liste */
int   time_b4(buf, p_next)
char  *buf;                /* Zeile der DIR Ausgabe */
struct s_dir_node *p_next;/* Zeiger auf den Vergleichsknoten */
{
   int   rtn;              /* Rückgabe von strncmp() */

/* Vergleich von Jahr, Monat, Tag, am/pm, Stunde und Minute */

   if(rtn = strncmp(&buf[29], &(p_next->dir_line)[29], 2))
      return(rtn < 0);                     /* Unterschied Jahr */
   if(rtn = strncmp(&buf[23], &(p_next->dir_line)[23], 2))
      return(rtn < 0);                    /* Unterschied Monat */
   if(rtn = strncmp(&buf[26], &(p_next->dir_line)[26], 2))
      return(rtn < 0);                      /* Unterschied Tag */
   if(buf[38] != (p_next->dir_line)[38])/* Unterschied am/pm */
      return(buf[38] == 'a');
   if(rtn = strncmp(&buf[33], &(p_next->dir_line)[33], 2))
      return(rtn < 0);                  /* Unterschied Stunden */
   if(rtn = strncmp(&buf[36], &(p_next->dir_line)[36], 2))
      return(rtn < 0);                   /* Unterschied Minute */
   return(0);              /* Datum und Uhrzeit sind gleich */
}
```

Listing 19: `TIME_B4.C.`

```
/*****************************************************************
 * SHO_LIST.C: Anzeige einer verketteten Liste
 *             Ausgabe nach stdout
 */

#include <stdio.h>          /* Für NULL Zeiger */
#include "by_time.h"        /* Struktur für verkettete Liste */


sho_list(p_head)
struct s_dir_node *p_head;/* Zeiger auf Listenbeginn */
{
   struct s_dir_node *p_next; /* Zeiger auf nächsten Knoten */

   for(p_next = p_head->next;  /* Start beim ersten Knoten */
       p_next != NULL;            /* Weitere Elemente? */
       p_next = p_next->next)     /* Nächster Knoten */

      puts(p_next->dir_line);     /* Ausgabe der Zeile */
}
```

Listing 20: `SHO_LIST.C.`

```
/*****************************************************************
 * DANGLE.C: Gefahr beim Referenzieren eines Zeigers auf
 *           deallokierten Speicher
 */
main()
{
   char  *p_text;

   p_text = str_dang();
   puts(p_text);
   printf("%s\n", p_text);
   printf("%s\n", p_text);
}

str_dang()
{
   char  auto_string[10];

   strcpy(auto_string, "123456789");
   return(&auto_string[0]);
}
```

Listing 21: `DANGLE.C` - *hüten Sie sich vor wilden Zeigern.*

Eine Programmier-Anleitung:

Ihr erster MS-DOS Gerätetreiber

»Das Programmieren eines Gerätetreibers ist eine schwierige, aufwendige Angelegenheit, die nur Betriebssystemgurus wagen sollten.« Wer dieses Vorurteil hat, hat einfach unrecht, denn das Programmieren eines Gerätetreibers ist weder sehr unterschiedlich vom Schreiben eines anderen Programms, noch ist es wesentlich schwieriger.

Dieser Artikel beleuchtet das Innenleben eines Gerätetreibers und bietet einige nützliche Tips und Techniken zum Debuggen eines Gerätetreibers. Der hier beschriebene Gerätetreiber MDM_DRV dient als einfaches Kommunikationsprogramm mit Xmodem-Prüfsummen-Protokoll. Obwohl es dasselbe Programm an der anderen Seite der Leitung oder des Modems erwartet, kann es auch Dateien aus einigen Mailboxen laden.

Ein Gerätetreiber ist das logische Interface zwischen dem Ein-/Ausgabe-System des Betriebssystems und der darunterliegenden Hardware. In einigen Fällen ersetzt der Gerätetreiber Teile des BIOS oder erweitert sie, und muß daher die Hardware direkt ansteuern. In anderen Gerätetreibern wird die physikalische Ein-/Ausgabe nur vom BIOS durchgeführt. Frühere DOS-Versionen erlaubten keine einfache Installation von Gerätetreibern. Seit der Version 2.0 kann der Programmierer aber das Betriebssystem leicht erweitern. Da die Systemschnittstelle für Gerätetreiber klar definiert ist (im Gegensatz zu Hintergrundprogrammen, TSRs), »vertragen« sich die Gerätetreiber in der Regel mit ihren Artgenossen.

Gerätetreiber gibt es in zwei Ausführungen: zeichen- und blockorientiert. Gerätetreiber, die das Speichern und/oder Lesen von Daten auf Disketten oder anderen Speichermedien mit wahlfreiem Zugriff erlauben, sind gewöhnlich Block-Gerätetreiber. Sie übertragen Daten in festen Blockgrößen, daher auch ihr Name.

Dieser Artikel beschränkt sich auf Zeichen-Gerätetreiber. Zeichen-Gerätetreiber müssen in einem von zwei Modi arbeiten: *cooked* oder *raw*. Der Cooked-Modus führt einfach Anforderungen an den Gerätetreiber für jeweils ein Zeichen durch und bearbeitet besondere Fälle wie Carriage Return oder Line Feed, die übersetzt werden können oder nicht. Bei jeder Eingabe wird eine Statusüberprüfung durchgeführt, um zu erkennen, ob [Control][C] gedrückt wurde, und ein Gerätetreiber, der im Cooked-Modus arbeitet, erlaubt MS-DOS in diesem Fall, den Treiberaufruf abzubrechen. Da das aufrufende Programm nicht wissen muß, ob der Gerätetreiber im Cooked- oder im Raw-Modus arbeitet, muß MS-DOS die typische Ein-/Ausgabe-Anfrage für mehrere Byte vom Programm in die richtige Anzahl Aufrufe für Einzelbytezugriffe umsetzen. MS-DOS erledigt dies unter Zuhilfenahme von Puffern, die nach Abarbeiten des Aufrufs übertragen werden.

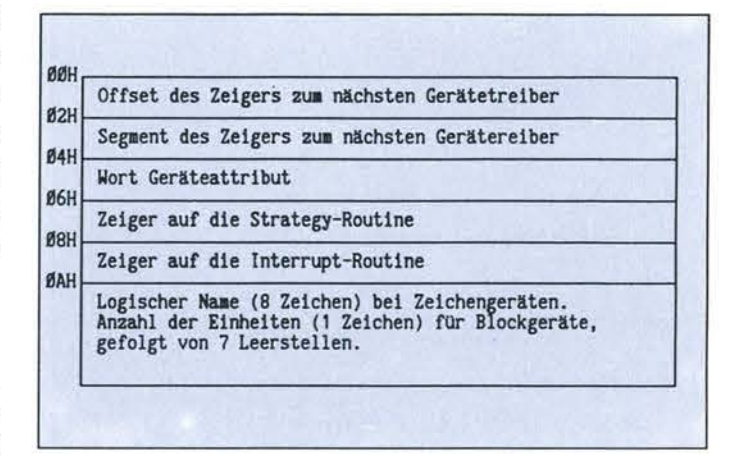


Bild 1: *Header eines Gerätetreibers.*

Im Raw-Modus wird keine Übersetzung oder spezielle Bearbeitung durchgeführt, weder vom Gerätetreiber, noch von MS-DOS. Im Raw-Modus gibt es keine Einzelbytezugriffe, der Gerätetreiber muß in der Lage sein, eine Anforderung für eine bestimmte Anzahl von Byte zu erledigen. Diese Zugriffe benötigen in der Regel keine von MS-DOS bereitgestellten Puffer, sondern werden in Puffern des Anwendungsprogramms durchgeführt. Deshalb ist der Raw-Modus effizienter, als der Cooked-Modus.

Der Anwender kann den Modus von raw nach cooked und umgekehrt ändern, indem er den Funktionsaufruf IOCTL (Interrupt 21H mit AH=44) durchführt.

Die Struktur der Gerätetreiber

Die drei Hauptteile des Gerätetreibers sind der *Header*, die *Strategy*- und die *Interrupt-Routine*. Der Header beschreibt die Fähigkeiten und Attribute des Treibers, benennt den Zeichen-Gerätetreiber, besitzt NEAR-Zeiger (nur Offset) auf die Strategy- und Interrupt-Routine und FAR-Zeiger (Segment und Offset) auf den nächsten Gerätetreiber in der Kette der Treiber. Diese Kette enthält nur Vorwärtsverweise, was Probleme beim Auffinden des Anfangs verursacht. Der Zeiger auf den nächsten Treiber wird von MS-DOS unmittelbar nach dem Ausführen der Initialisierungsroutine gesetzt, und muß anfänglich im Treiber auf -1 (FFFFFFFFH) gesetzt werden.

Der Gerätetreiber darf maximal 64 Kbyte groß sein, da die Zeiger auf die Strategy- und Interrupt-Routinen nur in ein Segment zeigen können. Den Header sehen Sie in *Bild 1* und eine Beschreibung der Bits im Attributwort in *Bild 2*.

Die Strategy-Routine

Die Strategy-Routine wird beim Installieren während des Bootvorgangs und bei jedem vom Betriebssystem generierten Aufruf aufgerufen. Eine einzige Ein-/Ausgabe-Anforderung des Anwendungsprogramms kann mehrere Ein-/Ausgabe-Anforderungen für den Treiber erzeugen.

Bit	Beschreibung
15	0 falls Blockgerät 1 falls Zeichengerät
14	0 IOCTL wird nicht unterstützt 1 IOCTL wird unterstützt
13	0 (falls Bit 15 0 ist) IBM-Format 1 nicht IBM-Format 0 (falls Bit 15 1 ist) Ausgabe bis Busy wird nicht unterstützt 1 Ausgabe bis Busy wird unterstützt
12	0 Undefiniert
11	0 Nur DOS 2.x Aufrufe 1 DOS 3.x. Aufrufe. Aufrufe für Open/Close Device und wechselbares Medium werden unterstützt. Diese werden bei DOS 2.x ignoriert.
10	0 Undefiniert
9	0 Undefiniert
8	0 Undefiniert
7	0 Undefiniert
6	0 Undefiniert
5	0 Undefiniert
4	0 Undefiniert
3	0 Normales Gerät 1 Spezial Uhr-Gerätetreiber
2	0 Normales Gerät 1 Null-Gerätetreiber
1	0 Normales Gerät 1 Standard-Ausgabegerätetreiber
0	0 Normales Gerät 1 Standard-Eingabegerätetreiber

Bild 2: *Attributwort des Gerätetreibers.*

Zweck der Routine ist das Speichern der Adresse für die spätere Bearbeitung und die anschließende Rückkehr. Die im Registerpaar ES:BX übergebene Adresse zeigt auf eine Struktur, die *Request Header* (*Listing 1*) genannt wird. Diese Struktur enthält Informationen für den Gerätetreiber, um ihm mitzuteilen, was er auszuführen hat.

Wichtig ist, daß keine Ein-/Ausgabeoperationen von der Strategy-Routine durchgeführt werden und daß die Adresse des Request-Headers vom Treiber für die spätere Bearbeitung abgespeichert wird. In einem echten Multitasking-System würde diese Adresse wahrscheinlich in einem Array gespeichert und nach einer Methode, die die optimale Geräte-Ausnutzung garantiert, sortiert. MS-DOS führt unmittelbar auf den Strategy-Routinen-Aufruf den Interrupt-Routinen-Aufruf durch. Ein wichtiger Punkt ist, daß die Interrupts zwischen dem Strategy- und dem Interrupt-Routinenaufruf nicht gesperrt sind: Das kann Probleme verursachen, wenn Ihr Gerätetreiber davon ausgeht, daß zwischen beiden Routinenaufrufen keine Verzögerung vorhanden ist.

```
rlength db  0         ; 0 - Länge der Daten im Header
unit    db  0         ; 1 - Gerätenummer (nicht in MDM_DRV)
command db  0         ; 2 - Das aktuelle Kommando
status  dw  0         ; 3 - Rückkehr-Status
reserve db  8 dup(0); 5 - Reserviert für DOS
media   db  0         ;13 - Media-Beschreiber (nicht in MDM_DRV)
address dd  0         ;14 - Zeiger auf Ein-/Ausgabe
count   dw  0         ;18 - Vorzeichenloser Zähler für Zeichen-
                      ;     Ein/Ausgabe
sector  dw  0         ;20 - Startsektor (nicht in MDM_DRV)
```

Listing 1: *Aufbau des Request-Headers.*

Die Interrupt-Routine

In der Interrupt-Routine wird die eigentliche Arbeit erledigt, und deshalb ist sie der komplizierteste Teil des Treibers. Beim Aufruf dieser Routine wird das Kommandobyte (drittes Byte) des vorher gespeicherten Request-Headers untersucht und die entsprechende Aktion ausgeführt. Sie sehen in *Bild 3* eine Liste der Kommandos mit den zugehörigen Aktionen.

Die Interrupt-Routine benutzt normalerweise das Kommandobyte als Offset für eine Verteilertabelle und ruft so die entsprechende Funktion für das Kommando auf. Sie können natürlich auch eine Sprungtabelle anlegen, wenn Sie möchten. Der Request Header enthält alle Informationen für die Bearbeitung und informiert das aufrufende Programm (in der Regel MS-DOS) nach der Bearbeitung über den Status. Das Statuswort enthält eine Anzahl Felder (*Bild 4*). Es enthält ein Fehlerbit, um anzuzeigen, daß ein anderer Teil den Fehlercode enthält. Ein Done-Bit zeigt an, daß die Operation vollständig ausgeführt wurde. Ein Busy-Bit wird momentan verwendet, um den Zustand des Geräts anzuzeigen.

Normalerweise ist es nicht ausreichend, den Status der Operation zurückzugeben. Oftmals muß die Anzahl der bearbeiteten Zeichen zurückgegeben werden, so bei Lese- oder Schreiboperationen.

Obwohl die Interrupt-Routine nicht über einen Interrupt aufgerufen wird, sondern mit einem FAR CALL, sollte sie sich wie eine richtige Interrupt-Routine verhalten, und alle Register und die Flags retten und am Ende wieder herstellen. Das Routinenende muß ein FAR RET sein.

Die Kommandos

Die in Bild 4 dargestellten Kommandos sind in MDM_DRV als aufrufbare Unterprogramme implementiert. Die Sprungverteilertabelle unterhalb des Request Headers im Assemblerlisting zeigt auf die benötigten Routinen. Von der Interrupt-Routine wird auf die Tabelle zugegriffen, wobei das Kommandobyte des Request Headers als Index benutzt wird.

Kommando-byte	Bedeutung
00H	Initialisierung für den Gerätetreiber. Wird nur beim Bootvorgang bei der Treiberinstallierung aufgerufen.
01H	Media Check. Der Rückgabestatus gibt an, ob das Medium gewechselt wurde (in MDM_DRV nicht verwendet).
02H	Erstelle den BIOS-Parameter-Block. Wird bei neuem oder gewechselten Medium aufgerufen. Der BPB soll erstellt werden, und die Adresse zurückgegeben werden (in MDM_DRV nicht verwendet).
03H	Read IOCTL. Kopiere Gerätetreiber-Informationen in einen lokalen Puffer.
04H	Read. Lesen einer Anzahl von Zeichen.
05H	Non-Destructive Read. Lesen des nächsten Zeichens von dem Gerät, ohne es aus dem Puffer zu nehmen.
06H	Input Status. Rückgabe eines Status, der angibt, ob ein Zeichen im Eingabepuffer wartet.
07H	Input Flush. Leeren des Eingabepuffers des Geräts.
08H	Write. Ausgabe einer Anzahl von Zeichen.
09H	Write Verify. Ausgabe der Zeichen mit Überprüfung, ob sie korrekt ausgegeben wurde.
0AH	Output Status. Statusrückgabe, die angibt ob das Gerät beschäftigt ist oder frei.
0BH	Flush Output Buffers. Leeren des Ausgabepuffers des Geräts.
0CH	Write IOCTL. Kopieren einiger Informationen in die Gerätetreiber-Tabellen.
0DH	Device Open (nur DOS 3.x). Initialisieren des Geräts.
0EH	Device Close (nur DOS 3.x). Wird zum Schließen der File Handle aufgerufen.
0FH	Removable Media Routine (nur DOS 3.x). Nur Blockgeräte.
10H	Output until Busy (nur DOS 3.x). Erlaubt die Rückkehr, bevor die Ausgabe beendet ist.

Bild 3: *Die Bedeutung des Kommandobytes im Request-Header.*

15	0	Kein Fehler
	1	Fehler
14		Reserviert
13		Reserviert
12		Reserviert
11		Reserviert
10		Reserviert
9	1	Gerät Busy
8	1	Operation beendet. (Liegt eine Fehlerbedingung vor, wird dieses Bit ignoriert.
0-7	Spezifische Fehlercodes:	
	0H	Schreibschutz
	1H	Unbekanntes Gerät
	2H	Laufwerk nicht bereit
	3H	Unbekanntes Kommando
	4H	Datenfehler beim Lesen
	5H	Unbekannte Ein-/Ausgabe Requeststruktur
	6H	Fehler beim Seek
	7H	Unbekanntes Medium
	8H	Sektor nicht gefunden
	9H	Drucker Fehler (Papierende)
	AH	Schreibfehler
	BH	Lesefehler
	CH	Allgemeiner Fehler
	DH	Reserviert
	EH	Reserviert
	FH	Mediumwechsel nicht erlaubt

Bild 4: *Statuswort eines Gerätetreibers.*

Sobald die Strategy- und die Interrupt-Routinen funktionieren, müssen die Kommandos implementiert werden. Dies ist eine geradlinige Vorgehensweise wie Sie im Listing sehen können. Alle unten detailliert beschriebenen Kommandos sind in derselben Reihenfolge vorhanden. Dieser Aufbau kann als Grundlage für fast alle Gerätetreiber dienen, unabhängig von ihrer Kompliziertheit. Zur weiteren Information über das Programmieren von Gerätetreibern können Sie auch das *MS-DOS Technical Reference* oder das Buch *MS-DOS für Fortgeschrittene* von Ray Duncan (Microsoft Press/Vieweg) heranziehen.

MDM_DRV benutzt nicht alle verfügbaren Kommandos, aber Sie werden aus Gründen der Vollständigkeit alle dargestellt. Die unbenutzten sind im Quellcode als einfache Unterprogramme mit einem Rücksprungbefehl dargestellt.

Jedes benötigte Kommando folgt derselben Logik. Es wird von der Interrupt-Routine mit einem NEAR CALL aufgerufen, führt seinen Zweck aus und gibt im Register AX einen Status zurück. Der Status wird mit dem Done-Bit logisch ODER verknüpft und dann im Statuswort des Original Request-Headers an der Stelle RH+3 gespeichert. Eventuell für diese Operation benötigte Zeichenzähler werden in der jeweiligen Routine aktualisiert. Wird eine Routine von diesem Treiber nicht durchgeführt, gibt sie einen Status zurück, der angibt, daß sie beendet wurde.

Die Initialisierungsroutine des Treibers

Die Initialisierungsroutine (Kommando 00) wird nur ein einziges Mal bei der Installation aufgerufen. Sie sollte das Gerät für die folgenden Operationen vorbereiten und kann eine Nachricht auf dem Bildschirm ausgeben. Zu Testzwecken in der Entwicklungsphase können Sie auch das Codesegment des Treibers ausgeben.

Sie können nur MS-DOS-Funktionsaufrufe mit Funktionscodes (im Register AH) kleiner oder gleich 0CH und die Funktion 30H zum Ermitteln der Versionsnummer durchführen. Da einige der Funktionen versionsspezifisch sind, werden Sie möglicherweise die Version ermitteln, ein Flag setzen, oder entsprechende Aktionen durchführen, wenn die falsche Betriebssystem-Version läuft.

```
com_port       dw  0         ; Nummer der seriellen Schnitt-
                             ; stelle. COM1 = 0, COM2 = 1 ...
init_data      db  10000011b ; Initialisierungsbyte
                             ; Details entnehmen Sie dem BIOS
                             ; Technischen Referenzhandbuch
block_num      dw  0         ; Aktuelle Blocknummer der
                             ; Übertragung
abort_xfer     dw  0         ; momentanen Transfer abbrechen
                             ; (True = Ja)
nak_cnt        dw  0         ; Anzahl erhaltener/gesendeter NAKs
cancel_flag    dw  0         ; Flag für CTRL-C
inrequest      dw  0         ; Anzahl nicht bearbeiteter Bytes
in_block       dw  0         ; Wurde ein Block empfangen?
was_dialed     dw  0         ; Modem wurde angerufen
```

Listing 2: *Die IOCTL-Struktur für MDM_DRV.*

Es gibt spezielle Operationen für Block-Gerätetreiber, die hier aber nicht behandelt werden, die statische Tabellen verwenden, so daß MS-DOS später die Attribute des Blockgeräts, die mit dem Treiber verknüpft ist, kennt.

Beim Aufruf der Initialisierungsroutine des Treibers zeigt der Doppelwort-Zeiger an der Stelle RH+18 auf die Stelle hinter dem Gleichheitszeichen in der Befehlszeile aus der Datei CONFIG.SYS. So können Sie auch Aktionen entsprechend von Angaben in der Kommandozeile ausführen. MDM_DRV zeigt diesen String am Bildschirm an.

Die oberste Speicheradresse, die vom Treiber belegt wird, wird im Doppelwort an der Stelle RH+14 zurückgegeben. Diese Information teilt MS-DOS mit, wo es den nächsten Gerätetreiber laden kann. Da die Initialisierung nur einmal aufgerufen wird, kann zur optimalen Speicherausnutzung die Adresse dieser Routine selbst zurückgegeben werden. Bedenken Sie, daß alles im Speicher hinter dieser Adresse überschrieben wird.

MDM_DRV benutzt zwei Funktionen, die nur im MS-DOS der Version 3.x und höher verfügbar sind, die Routinen *Device Open* und *Device Close*. Frühere Versionen von MS-DOS wußten nichts von der Existenz dieser Routinen. Da diese benötigt werden, veranlaßt eine falsche Betriebssystem-Version die Beendigung des Treibers.

MDM_DRV benutzt die Initialisierungsroutine, um sich zu vergewissern, daß es auf einem MS-DOS-System der Version 3.x läuft. Ist dies nicht der Fall, erfolgt die Ausgabe einer Fehlermeldung und der Abbruch. Sonst erfolgt die Initialisierung des Modem-Ports auf 1200 Baud, 8 Datenbits, keine Parität und ein Stopp-Bit.

Es gibt Alternativen zum Abbruch bei der falschen Version von MS-DOS. Mit etwas mehr Aufwand wäre es möglich, die Open-Funktion den ersten Lese- oder Schreibzugriff ausführen zu lassen. Der letzte Aufruf oder eine entsprechende Pause nach einer Rückkehr von einem Lese- oder Schreibaufruf, könnte die Funktion Device Close aufrufen. Ich beschloß, den Treiber nicht für die Version 2.x zu programmieren, schließlich ist es langsam Zeit, auf eine neuere Version umzustellen. Zuletzt wird eine Nachricht ausgegeben, und die Initialisierung ist abgeschlossen.

Gerät öffnen

Die Funktion Device Open (Kommandocode 0DH) hat keine definierte Funktion unter MS-DOS, aber Sie können sicher sein, daß MS-DOS Version 3.x diese vor jeder anderen Funktion aufruft, allerdings unter der Voraussetzung, daß das entsprechende Bit (Bit 11) im Header des Attribut-Blocks gesetzt ist. Der Treiber MDM_DRV benutzt dieses Bit für verschiedene Aufgaben.

Beim ersten Aufruf erfolgt die Initialisierung des Kommunikationsports mit den voreingestellten Werten. Diese Einstellung kann später mit dem Aufruf IOCTL geändert werden. Der Data Carrier Status wird überprüft, um zu erkennen, ob der Port aktiv und mit einem anderen Computer verbunden ist. Wenn eine Verbindung besteht, wird die folgende Ruf- und/oder Antwortaktion übergangen. Ist der Computer nicht Online, wird eine Telefonnummer oder ein [Return] vom Benutzer erwartet. Wurde eine Nummer eingegeben, so wählt das Modem, ansonsten erwidert es den Anruf und wartet auf den Trägerton. [Ctrl][C] führt zum sofortigen Abbruch.

Nach Entdecken des Trägers wird eine sehr einfache Terminal-Kommunikationsroutine aufgerufen. Diese gibt an der Tastatur eingegebene Zeichen sowohl auf dem Bildschirm, als auch am Kommunikationsport aus. Zeichen vom Kommunikationsport werden auf dem Bildschirm ausgegeben. Nach der Eingabe des Zeichens Escape auf einer der beiden Seiten verlassen beide Seiten die Routine und fahren fort. Nach der Eingabe von Escape wird der Timer in Beschlag genommen, da das Zeitverhalten ein kritischer Punkt des Xmodem-Protokolls ist. Am Schluß wird die Kontrolle dem ursprünglichen Anrufer übergeben.

Lesen

Das Kommando READ (Kommandocode 04H) ist bei weitem die schwierigste Routine in MDM_DRV.

Die aufrufende Routine (in den meisten Fällen MS-DOS) ruft die Funktion mit einer beliebigen Anzahl zu lesender Zeichen auf. Der Zeichenzähler ist im Request-Header an der Stelle RH+18. Die Zeichen werden an das aufrufende Programm zurückgegeben, indem sie sequentiell ab der Adresse, die an der Stelle RH+14 steht, gespeichert werden.

Xmodem überträgt aber die Daten in festen, 128 Byte großen Blöcken mit einigen zusätzlichen Protokoll-Informationen (sehen Sie auch den gesonderten Teil über Xmodem). Es erwartet entweder das Zeichen ACK (Acknowledge) oder NAK (negatives Acknowledge) innerhalb einer bestimmten Zeit von der empfangenden Stelle. Hiervon kann abgeleitet werden, ob der Block richtig empfangen wurde. Stellen Sie sich vor, eine zeichenweise Übertragung findet statt. Für den ersten Aufruf steht noch kein Zeichen bereit, so muß er auf die Übertragung eines Blocks warten. Wurde dieser Block nicht richtig übertragen, wenn also die Blocknummer, deren Komplement oder das Prüf-

summenbyte nicht so sind wie erwartet, oder wenn ein Timeout auftritt, dann wird ein NAK zurückgegeben und die Übertragungsschleife nochmal durchgeführt. Wurde der Block richtig übertragen, kann das ACK nicht gesendet werden, bis der Treiber bereit ist, einen neuen Block zu empfangen. Wenn das Programm TSRCOMM (auf Diskette) zur Verfügung steht, kann das ACK gesendet werden. (TSRCOMM erlaubt asynchrone Interrupts, wodurch ankommende Zeichen richtig empfangen werden.) Ist jedoch kein asynchroner Interrupt-Handler vorhanden, gehen Zeichen verloren, wenn das ACK unmittelbar gesendet wird. Es wird nicht gesendet, bevor die Zeichen im Block richtig verarbeitet und an der entsprechenden Stelle im Puffer des Aufrufers gespeichert sind. Das bedeutet, daß die folgenden Aufrufe mit der Kenntnis, daß die Zeichen bereits im Puffer sind, durchgeführt werden müssen.

Hierbei ergibt sich das Problem, daß es länger dauern kann, bis die 128 Zeichen durch die Treiberaufrufe bearbeitet sind, als zwischen dem Erhalten des letzten Zeichens und der Ausgabe von ACK oder NAK erlaubt ist.

Wie sieht es in dem Fall aus, wenn viele Zeichen angefordert werden? Die Leseroutine muß wieder auf die Ankunft eines Blocks warten und dann so viele Zeichen bearbeiten, wie im Zeichenzähler im Request Header angegeben sind, bevor sie zum aufrufenden Programm zurückkehrt. Verlangt der Leseaufruf mehr Zeichen als im Puffer sind, so läuft die Routine in der Schleife »Block lesen/ Zeichen verarbeiten«, bis alle Zeichen übertragen sind. Die Routine aktualisiert auch den Zeichenzähler.

Nach Beendigung der Leseanforderung wird die Kontrolle wieder an den Aufrufer übergeben. Gab es zu viele NAKs, und der Transfer wurde abgebrochen, wird die Kontrolle mit gesetztem Fehlerstatusbit abgegeben. Die Kontrolle wird mit einem Fehlercode zurückgegeben, wenn ein EOT (Ende der Übertragung) empfangen wurde oder wenn Leseanforderungen nach einem EOT erfolgen.

Eingangsstatus

Das Kommando INPUT STATUS (Kommandocode 06H) ist für diesen Treiber nicht besonders nützlich, aber dennoch implementiert. Wenn ein Zeichen verfügbar ist (ein Block wurde empfangen, aber nicht vollständig bearbeitet), wird ein Status von 0 im Statusbit des Statusworts an der Stelle RH+3 zurückgegeben. Außer bei Leseanforderungen in Schritten von 128 Byte gibt diese Routine normalerweise nach dem ersten Lesen den Status »Zeichen bereit« zurück.

Lesen ohne Entfernen

Das Kommando NON DESTRUCTIVE READ (Kommandocode 05H) übergibt nur das Zeichen, das beim nächsten Lesen zurückgegeben wird. Es wird an der Stelle RH+13 zurückgegeben, aber nicht aus dem Eingabepuffer entfernt. Was erfolgen soll, wenn der Eingabepuffer bei diesem Funktionsaufruf leer ist, ist nicht definiert. Daher wird ein Fehlercode von 0BH (Lesefehler) im Statuswort an der Stelle RH+3, und ein Zähler von 1 an der Stelle RH+18 zurückgegeben. Als Zeichen wird das Fragezeichen zurückgegeben.

Eingabepuffer leeren

Das Kommando FLUSH INPUT BUFFER (Kommandocode 07H) leert beim Aufruf den Eingabepuffer. Wird darauffolgend ein Leseaufruf durchgeführt, kann unter Umständen ein Timeout erfolgen, der veranlaßt, daß ein NAK gesendet wird. Das NAK veranlaßt, daß der letzte Block erneut gesendet wird, und somit ist alles in Ordnung.

Schreiben

Das Kommando WRITE (Kommandocode 08H) ist sehr einfach, verglichen mit dem Lesen. Beim Schreibaufruf steht die Anzahl Byte im Request Header an der Stelle RH+18. Die Transferadresse ist im Doppelwort an der Stelle RH+14 enthalten. Die Zeichen werden sequentiell in den Ausgabepuffer übertragen. Enthält der Ausgabepuffer 128 Zeichen, wird die Routine send_block aufgerufen. Der Block wird gesendet, bis er entweder erfolgreich übertragen wurde oder genügend NAKs oder Timeouts von der Gegenstelle empfangen wurden, und somit einen Übertragungsfehler ergeben.

Lesen mit Überprüfung

Das Kommando WRITE WITH VERIFY (Kommandocode 09H) macht für einen Zeichentreiber wenig Sinn und besonders wenig Sinn für einen Treiber, der das Xmodem-Protokoll verwendet. Normalerweise sollte die Schreibroutine aufgerufen werden, dann sollten die Daten zurückgelesen und verglichen, sowie bei Nichtübereinstimmung ein Fehler gemeldet werden. Im Falle des MDM_DRV genügt ein einfacher Aufruf der WRITE-Routine.

Ausgabestatus

Das Kommando OUTPUT STATUS (Kommandocode 0AH) soll das Busy-Bit setzen, wenn das Ausgabegerät beschäftigt ist. Da MDM_DRV weder gepuffert noch interruptgesteuert ist, ist dies möglich. Die Schreibroutine kehrt nie zurück, wenn nicht alle Aufgaben erledigt sind. Daher wird bei dem Rücksprung zum aufrufenden Programm das Busy-Bit nie gesetzt.

Ausgabepuffer leeren

Das Kommando FLUSH OUTPUT BUFFERS (Kommandocode 0BH) führt keine Aktionen in MDM_DRV aus, da das Leeren des aktuellen Ausgabepuffers schreckliche Auswirkungen auf die Transferroutine haben könnte.

Gerät schließen

Das Kommando DEVICE CLOSE (Kommandocode 0EH) wird wie das Kommando DEVICE OPEN nur unter MS-DOS 3.x und bei gesetztem Bit im Attributwort des Headers aufgerufen. Dieses Kommando überprüft, ob noch ausstehende Zeichen zu senden sind und erledigt dies mit Hilfe der Routine `send_block`. Wahrscheinlich wird dies der Fall sein, da die Ausgaberoutine nur jeweils ganze 128 Byte überträgt. Wird auf den abschließenden Block ein ACK empfangen, dann teilt die Routine der Gegenseite das Übertragungsende durch Senden von EOT mit.

Diese Routine wird auch beim Übertragungsabbruch oder bei der Eingabe von [Ctrl][C] aufgerufen. Daher wird das Abort-Flag geprüft. Ist es gesetzt, wird das Zeichen CAN gesendet, welches der Gegenstelle signalisiert, daß der Transfer abgebrochen wurde. In diesem Fall wird der Restblock nicht gesendet. War dies ein Anruf, wird der Hayes-kompatible Auflege-String gesendet.

Schließlich wird am Modem-Port DTR zurückgesetzt, die verbogenen Interruptvektoren werden wieder auf die ursprünglichen Werte gesetzt, und die Kontrolle geht an das aufrufende Programm.

Statusinformationen ans aufrufende Programm

Das Kommando I/O CONTROL READ (Kommandocode 03H) erlaubt zusammen mit dem Kommando I/O CONTROL WRITE die Übergabe von Statusinformationen an das aufrufende Programm. Es ist nicht voll definiert und kann deshalb für alle möglichen Fälle verwendet werden.

In MDM_DRV wird eine im *Listing 2* dargestellte Struktur übergeben. Diese Information erlaubt es dem Anwendungsprogramm zum Beispiel, den Modemport zu definieren, den MDM_DRV benutzt. Das Anwendungsprogramm kann auch erkennen, auf welche Werte diese Parameter augenblicklich gesetzt sind.

Bedenken Sie, daß die Zahl in RH+18 angibt, wie viele Zeichen Sie auf diesen Aufruf zurückgeben dürfen. Da bei diesem Aufruf normalerweise ein Puffer mit begrenzter Länge verwendet wird, kann die Rückgabe von mehr Zeichen an die Stelle in RH+14 das Schreiben in nicht korrekt allokierten Speicher bedeuten. In der Regel ist dies ein direktes Schreiben in einen Puffer des Anwendungsprogramms, wobei ein Aufruf durchaus nur das erste Wort (Port der Kommunikation), anstatt der ganzen Struktur erhalten will.

Informationen im Treiber setzen

Das Kommando I/O CONTROL WRITE (Kommandocode 0CH) erlaubt Ihnen auch Information an/vom Gerätetreiber zu übertragen. Dieses Kommando setzt Informationen innerhalb des Gerätetreibers.

Die normale Benutzung ist ein Aufruf I/O CONTROL READ in einen lokalen Puffer, das Ändern der gewünschten Daten und dann das Aktualisieren mit einem I/O CONTROL WRITE Aufruf.

Kommandos mit Dummyroutine

Die Kommandos CHECK MEDIA (Kommandocode 01H), BUILD BIOS PARAMETER BLOCK (Kommandocode 02H), REMOVABLE MEDIA (Kommandocode 0FH) und OUTPUT UNTIL BUSY (Kommandocode 10H) werden vom Treiber MDM_DRV nicht verwendet, sind aber aus Gründen der Vollständigkeit im Treiber enthalten. In der Sprungverteilertabelle wird die gemeinsame Adresse einer Dummyroutine verwendet.

Design-Überlegungen

Es gibt Probleme bei der Gestaltung und dem Programmieren des Gerätetreibers, die anders geartet sind, als bei einem normalen Programm. In vielen Belangen sind der Entwurf und die Implementierung eines Gerätetreibers ähnlich einer Windows-Anwendung: Ihre Hauptroutinen werden von anderen Tasks aufgerufen, auch öfters; das System arbeitet nicht weiter, bis Sie die Kontrolle abgeben; die Schnittstelle ist starr definiert.

Das erste Problem besteht darin: Es gibt keine Garantie, falls ein Device-Open-Aufruf durchgeführt wurde, daß auch ein Device-Close-Aufruf erfolgt. Um die Sache ein wenig verwirrender zu gestalten: Das einfache MS-DOS-Kommando COPY führt einige Device-Open- und Device-Close-Aufrufe aus, bevor es das Device für die Ein-/Ausgabe öffnet. (Die wirkliche Sequenz kann Device Open, Device Close, Device Open, Lesen/Schreiben und Device Close sein.)

Ein weiteres Problem des Gerätetreibers ist der Wiederholungszähler. Irgendwo in den Tiefen von MS-DOS ist eine Zahl, die angibt, wie viele Wiederholungen Sie wünschen, bevor der Gerätetreiber einen Fehler zurückgibt, und MS-DOS die Meldung »Abort, Retry, Ignore?« ausgibt. Dies wird vom Interrupt 24H, dem kritischen Interrupt-Handler erledigt. Es gibt keine dokumentierte Methode, nur *einen* Funktionsaufruf Device Open aufzurufen, und anschließend sofort den Interrupt 24H. Deshalb gibt es keinen sauberen Weg, wegen einer »toten Leitung« abzubrechen.

Da MS-DOS aus irgendeinem Grund außerhalb der Initialisierungsphase nicht vom Treiber aufgerufen werden darf, kann nur das BIOS für Zeichen-Ein-/Ausgabe aufgerufen werden. Direktes Lesen von der Tastatur und direktes Schreiben in den Bildschirmspeicher funktionieren auch, erschweren aber die ganze Angelegenheit. Das bedeutet aber auch, daß ein Abbruch mit [Ctrl][C] nicht so funktioniert wie Sie sich das wünschen, um vom Device Open Aufruf zum aufrufenden Programm bzw. MS-DOS zurückzukehren oder den kritischen Interrupt aufzurufen.

Dies ist ein weiteres schwieriges Problem, da MS-DOS-Gerätetreiber nicht beabsichtigen, ein interaktives Programm zu sein. Dies ist eine Herausforderung, die ich bis jetzt nicht richtig gelöst habe. Wenn Sie einmal den Treiber aufgerufen haben, gibt es keine Umkehr mehr.

Das ist das kleinste Problem, wenn jemand mit MS-DOS Dinge unternimmt, für das es nicht gedacht ist. Dagegen mußte ein Problem durch ein externes Programm gelöst werden: MS-DOS öffnet Zeichen-Gerätetreiber immer im Cooked-Modus. Wenn Sie eine Datei übertragen, und diese Datei ein CTRL-Z enthält, wird dieses als Dateiendezeichen verstanden, und die Datei wird sofort geschlossen. Deshalb muß der MDM_DRV im Raw-Modus arbeiten. Wenn Sie es nicht selbst vorsehen, wird nicht garantiert, daß nach jedem Open-Aufruf auch ein IOCTL-Aufruf stattfindet, um den Treiber in den Raw-Modus zu schalten.

Die Umstellung in den Raw-Modus hält nur für die Lebensdauer einer File Handle. Die weiteren Open-Aufrufe sind wieder im Cooked-Modus, bis Sie das kleine Hintergrundprogramm SET_MDM laufen lassen. Dieses Programm fängt einfach den MS-DOS Interrupt 21H ab und prüft, ob ein Open mit File-Handle-Aufruf vorliegt. Ist dies der Fall, und die Adresse im Registerpaar DS:DX zeigt auf MDM (gefolgt von einer NULL), wird der Filemodus nach einem erfolgreichen Aufruf auf den Raw-Modus gesetzt. Dies ist für das aufrufende Programm transparent und erlaubt - wenn auch nicht besonders schön.

Jeder Programmierer sollte ein vernünftiges Programm haben, mit dem er seine eigenen vergleichen kann. Ich schlage SET_MDM als neuen Standard vor. Aber wie immer, wenn Ihres funktioniert, ist dies in Ordnung.

Verbesserungen

Da MDM_DRV ein Beispiel für einen funktionierenden Gerätetreiber ist, ist der Code nicht ausgefeilt, und es gibt Möglichkeiten für die Verbesserung.

Die Routine `get_num` prüft zum Beispiel nicht, ob das eingegebene Zeichen ein erlaubtes ASCII-Zeichen ist. Sie wartet nur auf ein Return oder bis zu 20 Zeichen. Eine gute Möglichkeit, diese Routine (oder jede andere im Treiber) abzubrechen, steht ganz oben auf meiner Liste für Verbesserungen. Des weiteren erwartet der Treiber ein Hayes-kompatibles Modem an einem COM-Port. Diese Annahme ist fest in den Aufrufen `dev_open` und `dev_close` kodiert. Wird MDM_DRV für andere Modems oder mit anderen Geräten, die an die serielle Schnittstelle angeschlossen sind, verwendet, so ist ein zusätzliches Programm notwendig - möglicherweise auch die Funktionen IOCTL.

Das Protokoll ist einfach die Xmodem-Prüfsumme. Eine weitere Verbesserung wäre die Verwendung der verbesserten Xmodem-CRC-Berechnung. Die Implementierung der Verbesserungen erfordert keinen großen Aufwand.

Ein Abbruch oder ein EOT wird nicht entsprechend der Spezifikation erledigt, da sofort abgebrochen wird. Die Spezifikation verlangt, daß das erste CAN oder EOT mit einem NAK beantwortet wird, um sicherzustellen, daß es sich nicht um Leitungsrauschen handelte.

Debuggen

Die Entwicklung eines Programms erfordert Geduld und detektivisches Vorgehen, wenn ein Problem auftritt. Es ist aber schwierig, Gerätetreiber ohne entsprechende Hilfsmittel zu testen. Es stehen aber keine geeigneten Hilfsmittel zur Verfügung, Sie müssen sich leider eigene schaffen.

Das erste ist eine aktuelle Sicherungskopie Ihrer Platte und die Möglichkeit, das System ohne automatisches Laden des Gerätetreibers neu zu starten. Meine Methode ist eine Startdiskette mit `CONFIG.SYS`. Ich starte den Computer mit dieser Diskette, wenn ich den Treiber teste. Es folgen noch einige Vorschläge für das Erstellen von MDM_DRV:

- Erstellen Sie den Gerätetreiber zuerst als normales COM-Programm. Schreiben Sie sich dann einen Simulator, der alle Funktionen aufruft und mit einem Debugger die Register sowie den Request-Header ansehen läßt. Das ist zwar nicht ganz ideal, da es Ihnen keinen Test unter den wirklichen Voraussetzungen erlaubt, es ist aber ein guter Start und erlaubt Ihnen den Übergang zur nächsten Methode.
- Schaffen Sie ein Programmskelett, das noch keine Initialisierungsroutine beinhaltet. Die Interruptroutine sollte nur ein Registerpaar mit der Adresse des Request-Headers laden, die Sie in der Strategy-Routine abgespeichert haben. Rufen Sie dann einen Interrupt auf, den Sie für sich reserviert haben. Sehen Sie hier ein Beispiel:

```
interrupt proc far
     push      es
     push      bx
     mov       es, cs:[old_segment]
     mov       bx, cs:[old_offset]
     int       60h
     pop       bx
     pop       es
     ret
interrupt endp
```

- Das setzt natürlich voraus, daß Sie eine Interruptroutine für diesen Interrupt installiert haben. Ich verwende Interrupt 60H, einen Interrupt für Anwender. Folgendes Programm können Sie vervollständigen:

```
int60 proc far
      call real_int_routine
      iret
int60 endp
```

Das erlaubt Ihnen, `real_int_routine` als eine Far-Routine mit einem FAR RET aufzurufen. Der Vorteil dieser Vorgehensweise ist, daß Sie ein einfaches TSR-Programm schreiben können, das den Interrupt übernimmt und sich verhält, als wäre es ein Gerätetreiber.

Wenn Sie es als TSR-Programm getestet haben, was mit Debuggern leicht möglich ist, können Sie den Code in den wirklichen Gerätetreiber ändern und sind bis auf die Initialisierung fertig.

- Der hardwareunterstützte Debugger Periscope gestattet es, als `public` deklarierte Variablen irgendeinem Codesegment zuzuordnen. Wenn Ihre Initialisierungsroutine beim Starten das Codesegment anzeigt, können Sie auf einfache Weise ein Dummyprogramm erstellen, das mit der Symboltabelle geladen wird, die Segmentadressen tauschen und den Treiber symbolisch debuggen. Ab diesem Zeitpunkt debuggen Sie quasi ein normales Programm.
- Beachten Sie, daß dem Gerätetreiber nur ein sehr kleiner Stackbereich zur Verfügung steht. Es gibt keine Möglichkeit festzustellen, wieviel Speicher dem Gerätetreiber wirklich zur Verfügung steht. Der beste Weg ist das Speichern des Original-Stacksegments und des Stackpointers, das Benutzen eines eigenen Stacks und das Wiederherstellen des Originalstacks am Routinenende. MDM_DRV macht dies in der Interruptroutine.
- Programmieren Sie den Gerätetreiber, als wäre er eine Hardware-Interruptroutine. Alle Register (und die Flags) müssen erhalten bleiben. Kritische Teile des Treibers sollten mit gesperrten Interrupts arbeiten (CLI).
- Vergewissern Sie sich, daß der Zeichenzähler die Anzahl der wirklich bearbeiteten Zeichen wiedergibt. Falsche Zählerangaben haben sonderbare Auswirkungen.
- Stellen Sie sicher, daß der Gerätetreiber keinen Speicher oberhalb der Adresse anspricht, die Sie bei der Initialisierung zurückgegeben haben.

Ross M. Greenberg

```
;;  MDM_DRV.ASM
;;  Ein XMODEM-Gerätetreiber
;;  Programmiert von Ross M. Greenberg
;;  Es folgen Beispielteile
    .
    .
code    segment public  'CODE'
;;  Der Treiber ist eine große Prozedur, die 'driver'
;;  genannt wird.

driver  proc    far
    assume  cs:code,ds:code,es:code ; Die Segmentregister
                            ; müssen auf das Codesegment zeigen
    org 0                   ; Treiber muß bei Offset 0 beginnen
;;  Header Gerätetreiber

header1 dd  -1              ; Muß -1 sein. Wird von DOS gesetzt.
        dw  0e800h          ; Zeichengerät
                            ; IOCTL unterstützt
                            ; Ausgabe bis Busy
                            ; Open/Close/RM unterstützt
        dw  strat           ; Zeiger zur Strategy-Routine
        dw  ints            ; Zeiger zur Interrupt-Routine
        db  'MDM      '     ; Name des Treibers.
                            ; Linksbündig und 8 Zeichen lang
```

```
;;  Der Request-Header

request struct
rlength db  0               ; 0 - Länge der Daten im Header
unit    db  0               ; 1 - Einheitennummer (unbenutzt)
command db  0               ; 2 - Das Kommando
status  dw  0               ; 3 - Rückgabe-Status
reserve db  8 dup (0)       ; 5 - Reserviert für DOS
media   db  0               ; 13 - Media Beschreiber (unbenutzt)
address dd  0               ; 14 - Doppelwort Zeiger für E/A
count   dw  0               ; 18 - unsigned int für Zeichen E/A
sector  dw  0               ; 20 - Startsektor (unbenutzt)
request ends

;;  Die Verteiler-Tabelle

dispatch:
    dw  init                ; 0x00 - Treiber Initialisierung
    dw  media_chk           ; 0x01 - Media Prüfung (unbenutzt)
    dw  bld_bpb             ; 0x02 - Erstellen des BPB (unbenutzt)
    dw  rd_ioctl            ; 0x03 - IOCTL lesen
    dw  read                ; 0x04 - Lesen von 'Anzahl' Zeichen
    dw  nd_read             ; 0x05 - Nicht destruktives Lesen
    dw  inp_stat            ; 0x06 - Eingabestatus
    dw  inp_flush           ; 0x07 - Eingabepuffer leeren
    dw  write               ; 0x08 - Zeichen ausgeben
    dw  write_vfy           ; 0x09 - Schreiben mit Verifizierung
    dw  out_stat            ; 0x0A - Ausgabestatus
    dw  out_flush           ; 0x0B - Ausgabepuffer leeren
    dw  wrt_ioctl           ; 0x0C - IOCTL schreiben
    dw  dev_open            ; 0x0D - Device öffnen (DOS 3.x)
    dw  dev_close           ; 0x0E - Device schließen (DOS 3.x)
    dw  rem_media           ; 0x0F - Routine für austauschbares
                            ;        Medium (DOS 3.x)
    dw  out_busy            ; 0x10 - Ausgabe bis bereit (DOS 3.x)

;;  Strategy Routine
strat   proc    far
    mov word ptr cs:[rh_ptr], bx
    mov word ptr cs:[rh_ptr + 2], es
    ret
strat   endp
    .
    .
    .
    .
;;  Interrupt Routine

ints    proc    far
    cli
    mov cs:[old_stack], sp
    mov cs:[old_stack + 2], ss
    mov sp, cs
    mov ss, sp
    mov sp, offset cs:new_stack_end
    sti
    PUSHALL
    push    cs
    pop ds
    les di, cs:[rh_ptr]
    mov bl, es:[di.command]
    xor bh,bh
    cmp bx, MAX_CMD
    jle ints1
    mov ax, ERROR + UNK_COMMAND
    jmp ints2
ints1:
    shl bx, 1
    call    word ptr dispatch[bx]
    les di, cs:[rh_ptr]
ints2:
    or  ax, DONE            ; DONE-Bit setzen
    mov es:[di.status], ax  ; im Status der Struktur
    POPALL
    cli
    mov ss, cs:[old_stack + 2]
    mov sp, cs:[old_stack]
    sti
    ret
ints    endp
```

```
;; READ ROUTINE

read    proc    near
    DO_PRINT    msg_read
    mov ax, es:[di.count]
    mov cs:[inrequest], ax  ; Speichern des Anforderungszählers
    xor ax, ax              ; Request Header Count
    mov es:[di.count], ax   ; Ø setzen
    lds bx, es:[di.adress]  ; ds:bx zeigt auf Daten
top_read:
    cmp cs:[in_block], TRUE ; sind wir innerhalb des Blocks?
    jnz lp_it
    jmp read_loop           ; Ja
lp_it:
    mov cs:[_incnt], FALSE
    mov si, offset cs:[_soh]
    mov cs:[timer], TEN_SECONDS ; Setzen des Timers
    mov cs:[nak_cnt], FALSE     ; NAK-Zähler zurücksetzen
    cmp cs:[block_num], 1       ; Erstes Mal?
    jnz rd_blk                  ; Nein
    call    send_nak            ; Senden des ersten NAK
    STAT    LEFT_BRACKET        ; [ anzeigen
rd_blk:
    call    get_char            ; Irgendwelche Zeichen?
    jc  rd_blk2                 ; Nein
    cmp cs:[_incnt], FALSE      ; Erstes Zeichen?
    jnz nxt_char                ; Nein
    cmp al, SOH                 ; Erstes Zeichen SOH?
    jz  nxt_char                ; Ja -> speichern
    cmp al, ABORT               ; Übertragung beenden?
    jz  abort_it                ; Ja -> Abbruch
    cmp al, EOT                 ; Ende?
    jnz rd_blk                  ; Nein Zeichen wegwerfen
    call    send_ack            ; EOF bearbeiten
    STAT    RIGHT_BRACKET       ; ] ausgeben
    STAT    CR                  ; und Zeilenvorschub
    STAT    LF
    xor ax,ax
    ret


nxt_char:
    mov cs:[si], al             ; Zeichen speichern
    inc cs:[_incnt]             ; Zähler erhöhen
    inc si                      ; Zeiger erhöhen
rd_blk2:
    cmp cs:[_incnt], FULLBLKSIZE; Genügend Bytes
    jge chkblk                  ; für die Überprüfung
    call    eat_char            ; für CTRL-C
    cmp cs:[timer], FALSE       ; Time out?
    jnz rd_blk                  ; nicht -> erneuter Versuch
    STAT    DOT                 ; Punkt anzeigen
    jmp bad_blk2
bad_blk:
    STAT    QUESTION            ; Fragezeichen ausgeben
bad_blk2:
    call    send_nak            ; NAK senden
    cmp word ptr cs:[abort], TRUE; Abbruch?
    jnz rd_blk3                 ; Nein -> Timer zurücksetzen
                                ; erneut versuchen
abort_it:
    call    send_abort          ; Abbruchzeichen senden
    call    send_abort          ; nochmal
    STAT    EXCLAIM             ; beenden
    mov ax, 8ØØch               ; Fehlerrückgabe
    ret
rd_blk3:
    mov cs:[timer], TEN_SECONDS ; Timer zurücksetzen
    jmp rd_blk                  ; und nochmals versuchen
    ret
chkblk:
    mov ax, cs:[block_num]
    push    ax
    dec al                      ; temporär
    cmp al, cs:[_blk]           ; doppelter Block
    jnz real_blk                ; nein
    STAT    DUP_BLK             ; ja
    xor ax,ax
    mov cs:[_incnt], ax
    jmp ack_blk
real_blk:
    pop ax                      ; den aktuellen Block sichern
    cmp cs:[_blk1], al          ; Ist der Blockzähler ok?
    jnz bad_blk                 ; nein
    not al
    cmp cs:[_blk2], al          ; Block Komplement
    jnz bad_blk                 ; nein
    mov cx, BLKSIZE
    mov si, offset _buf
    call    do_chksum
    cmp cs:[_chksum], al
    jnz bad_blk                 ; Prüfsumme falsch
    mov cs:[in_block], TRUE     ; In Ordnung
    mov cs:[_inptr], offset cs:_buf; Zeiger setzen
    sub cs:[_incnt], FULLBLKSIZE - BLKSIZE
read_loop:
    mov si, cs:[_inptr]         ; Zeiger zurücksetzen
rd_loop:
    mov al, cs:[si]             ; Zeichen nehmen
    mov ds:[bx], al             ; und abspeichern
    inc bx                      ; beide Zeiger erhöhen
    inc si
    inc word ptr es:[di.count]  ; Zähler erhöhen
    dec cs:[_incnt]             ; Restliche Zeichen vermindern
    jnz stf_nxt_char            ; Weitere Zeichen
    inc cs:[block_num]
    STAT    STAR                ; Stern anzeigen
ack_blk:
    call    send_ack            ; ACK senden
    mov cs:[in_block], FALSE
    jmp top_read
stf_nxt_char:
    dec cs:[inrequest]          ; noch Zeichen?
    jnz rd_loop
    xor ax,ax                   ; keine mehr angefordert,
    ret                         ; kein Fehler
read    endp

;; WRITE ROUTINE

write   proc    near
    PUSHALL AX
    DO_PRINT    msg_write
    mov cx, es:[di.count]       ; Wie viele Zeichen?
    xor dx, dx                  ; Zeichen Zähler löschen
    lds bx, es:[di.adress]      ; ds:bx = Zeiger auf Daten
    mov si, cs:[_outptr]
wr_lp:
    mov al, ds:[bx]             ; Zeichen
    mov cs:[si], al             ; im Puffer speichern
    inc bx                      ; Zeiger erhöhen
    inc cs:[_outcnt]            ; Gesamtzeichen Zähler
    inc dx                      ; aktueller Zeichen Zähler
    cmp cs:[_outcnt], BLKSIZE   ; Block senden?
    jnz wr_ok                   ; jetzt nicht
    call    send_block          ; Block senden
    cmp cs:[abort_xfer], FALSE  ; Abbruch?
    jz  blk_ok                  ; nein, da Transfer ok
    call    send_abort          ; ja
    call    send_abort          ; 2x senden
    mov es:[di.count], FALSE    ; keine Zeichen gesendet
    mov ax, ERROR + WRITE_ERROR ; Schreibfehler markieren
    ret


blk_ok:
    mov cs:[_outptr], offset _buf; Zeiger zurücksetzen
    mov si, offset _buf         ; Register zurücksetzen
    mov cs:[_outcnt], FALSE     ; Zeichenzähler zurücksetzen


wr_ok:
    inc si                      ; Zeichenzeiger erhöhen
    loop    wr_lp
    mov es:[di.count], dx       ; Anzahl gesendeter Zeichen
    mov cs:[_outptr], si        ; Zeiger speichern
    xor ax,ax                   ; keine Fehler
    POPALL AX
    ret
write   endp
```

```
;; DEVICE OPEN ROUTINE

dev_open     proc     near
     PUSHALL AX
     DO_PRINT     msg_dev_open
     mov cs:[cancel_flag], FALSE
     mov cs:[_inptr], offset cs:_soh
     mov cs:[_incnt], FALSE
     mov cs:[_outptr], offset cs:_buf
     mov cs:[_outcnt], FALSE
     mov cs:[block_num], 1
     call     set_timer
     mov dx, cs:[com_port]          ; COM-Port initialisieren
     mov al, cs:[init_data]
     mov ah, Ø
     int 14h
eat_loop:
     mov dx, ds:[com_port]
     mov ah, 3                      ; Status lesen
     int 14h
     test     ah, 1                 ; Daten bereit
     jz  continue                   ; nein
     mov ah, 2
     int 14h                        ; Zeichen überlesen und
     jmp eat_loop                   ; wiederholen
continue:
     test     al, Ø8Øh              ; DCD an?
     jz  off_line
     jmp on_line
off_line:
     mov dx, offset cs:offline      ; Nachricht ausgeben
     mov ah, 9
     int 21h
     mov si, offset cs:numbuf
     mov cx, 19                     ; maximale Länge
     call     get_num
     cmp cs:[kb_len], Ø             ; Nur RETURN?
     jnz dial_it
     mov dx, offset cs:await
     mov ah, 9
     int 21h
     mov si, offset cs:answerstring
     call     out_string
     mov cs:[timer], FOREVER
     jmp offline_lp                 ; Warten auf Trägersignal
dial_it:
     mov dx, offset cs:dialing
     mov ah, 9
     int 21h
     mov si, offset cs:dialstring
     call     out_string
     mov si, offset cs:numbuf
     call     out_string
     mov si, offset cs:return
     call     out_string
     mov cs:[timer], ONE_MINUTE
     mov cs:[was_dialed], TRUE

offline_lp:
     mov ah, 3                      ; Online?
     mov dx, cs:[com_port]
     int 14h
     test     al, Ø8Øh              ; DCD an?
     jnz made_con                   ; ja
     call     eat_char              ; für CTRL-C
     cmp cs:[timer], Ø              ; Time-out?
     jnz offline_lp
abort_call:
     mov cs:[was_dialed], FALSE
     mov dx, offset cs:no_con
     mov ah, 9
     int 21h
     xor ax, ax
     mov es:[di.count], ax          ; Zähler auf Ø setzen
     mov ax, ERROR + GEN_FAILURE    ; Fehlerhaft markieren
     POPALL AX
     ret
made_con:
     mov dx, offset cs:con
     mov ah, 9
     int 21h
term_em_lp:
     call     get_char
     jc  get_term_char
     cmp al, ESCAPE
     jz  term_exit                  ; ESCAPE -> Ende
     mov ah, Ø2h
     mov dl, al
     int 21h
get_term_char:
     mov ah, Øbh
     int 21h                        ; Irgendwelche Zeichen
     or  al, al
     jz  term_em_lp                 ; nein
     mov ah, 1                      ; ja -> Zeichen einlesen
     int 21h
     mov ah, 1
     mov dx, ds:[com_port]
     int 14h
     cmp al, ESCAPE
     jz  term_exit                  ; ESCAPE -> Ende
     mov ah, Ø2h
     mov dl, al
     int 21h
     jmp get_term_char
term_exit:
     mov dx, offset cs:xfer         ; Statusmeldung
     mov ah, 9
     int 21h
on_line:
     xor ax, ax
     POPALL AX
     ret
dev_open     endp

;; DEVICE CLOSE ROUTINE

dev_close     proc     near
     DO_PRINT     msg_dev_close
     cmp cs:[cancel_flag], TRUE     ; Abbruch?
     jnz no_cancel                  ; nein
     mov ah, 1                      ; ABORT senden
     mov dx, cs:[com_port]
     mov al, ABORT
     int 14h
     jmp no_send                    ; normales Ende
no_ancel:
     cmp cs:[_outcnt], FALSE        ; noch übrige Zeichen
     jz  no_send                    ; nein
     call     send_block            ; ja -> Rest senden
     STAT     RIGHT_BRACKET
     STAT     CR
     STAT     LF
     mov cs:[_outptr], offset _buf; Zeiger zurücksetzen
     mov si, offset _buf            ; Register zurücksetzen
     mov cs:[_outcnt], FALSE        ; Zeichenzähler zurücksetzen
     inc cs:[block_num]             ; vorsichtshalber...
no_send:
     cmp cs:[block_num], 1          ; mindestens 1 Block gesendet?
     jz  no_blk_sent                ; nein
     mov ah, 1                      ; EOT senden
     mov dx, cs:[com_port]
     mov al, EOT
     int 14h
no_blk_sent:
     cmp cs:[was_dialed], TRUE      ; war ein Anruf?
     jnz no_hang
     mov cs:[timer], ONE_SECOND     ; eine Sekunde Pause
sil_lp1:
     call     eat_char              ; für CTRL-C
     cmp cs:[timer], FALSE          ; Pause vorbei?
     jnz sil_lp1
     mov si, offset cs:pulses
     call     out_string
     mov cs:[timer], ONE_SECOND     ; eine Sekunde Pause
sil_lp2:
     call     eat_char              ; für CTRL-C
     cmp cs:[timer], FALSE          ; Pause vorbei?
     jnz sil_lp2
     mov si, offset cs:hangup       ; aufhängen
```

```
    call    out_string
no_hang:
    mov cs:[was_dialed], FALSE
    mov cs:[cancel_flag], FALSE
    call    reset_timer
    ret
dev_close   endp

;;  Die folgenden Routinen stehen im kompletten Programm
;;  an dieser Stelle:
;;  NON-DESTRUCTIVE READ ROUTINE
;;  INPUT STATUS ROUTINE
;;  FLUSH INPUT BUFFER ROUTINE
;;  OUTPUT STATUS ROUTINE
;;  FLUSH OUTPUT BUFFERS ROUTINE
;;  READ I/O CONTROL ROUTINE
;;  WRITE I/O CONTROL ROUTINE
;;  SEND_BLOCK ROUTINE
    .
    .
;;  INITIALIZE DRIVER ROUTINE
init    proc    near
    DO_PRINT    msg_init
    mov ah, 030h                ; DOS Version bestimmen
    int 21h
    cmp ah, 3                   ; Version 3.x?
    jge okay_dos                ; ja
    mov dx, offset cs:wrong_dos ; Nachricht ausgeben
    mov ah, 9
    int 21h
endless_loop:
    cli
    jmp endless_loop            ; nicht besonders schön
okay_dos:
    mov dx, offset cs:greetings ; Nachricht ausgeben
    mov ah, 9
    int 21h
    push    ds
    mov ds, es:[di.count + 2]   ; Segment der Zeile der
                                ; CONFIG.SYS Zeile
    mov si, es:[di.count]       ; Offset
    call    output_chars
    pop ds
    mov dx, offset cs:end_greetings ; Nachricht ausgeben
    mov ah, 9
    int 21h
    mov word ptr es:[di.address], offset init
    mov word ptr es:[di.address + 2], cs
    xor ax, ax
    ret
init    endp


wrong_dos   db  '?? Der Treiber muß unter DOS 3.0'
            db  ' oder höher laufen',
            db  CR, LF, 'System angehalten! $'
greetings   db  CR, LF, LF, 'MDM_DRV ist installiert...',CR,LF
            db  'CONFIG.SYS Zeile ist : $'
end_greetings
            db  CR, LF, LF, LF, '$'

output_chars    proc    near
output_loop:
    mov dl, ds:[si]             ; Zeichen holen
    cmp dl, LF                  ; Ende?
    jnz outit                   ; nein
    ret
outit:
    mov ah, 2                   ; Zeichen ausgeben
    int 21h
    inc si
    jmp output_loop
output_chars    endp

driver  endp
code    ends
    end
```

Listing 3: *Beispielteile aus dem Gerätetreiber MDM_DRV.*

Das Xmodem-Protokoll: Die fehlerfreie Übertragung eines Datenblocks

Xmodem überträgt Daten in 128 Byte großen Blöcken, und zwar jeweils einen. Zusätzlich enthält jeder Block Informationen, um sicherzustellen, daß die Übertragung fehlerfrei ist. Die zusätzliche Information, Prüfsumme genannt, ist eine Berechnung der Daten des Blocks. Die Kalkulation wird bei der Übertragung durchgeführt und das Ergebnis mit dem Block übertragen. Die empfangende Stelle führt dies ebenso durch und vergleicht das Resultat mit dem der sendenden Stelle. Weichen die beiden Ergebnisse voneinander ab, weist der Empfänger den erhaltenen Block ab, da er weiß, daß er falsche Daten empfangen hat. Die Zurückweisung erfolgt durch die Mitteilung, daß der Block falsch empfangen wurde. Der Sender sendet daraufhin den Block nochmals. Treten zu viele Fehler auf, kann eine Seite der anderen mitteilen, daß der Transfer abgebrochen wird.

Das Xmodem-Protokoll ist ein vom Empfänger gesteuertes Protokoll. Der Sender wartet, bis der Empfänger einen Block anfordert. Die Anforderung gilt entweder dem nächsten oder dem letzten Block.

Am Blockbeginn steht SOH (ASCII 01H), gefolgt von einem Byte, das die Blocknummer angibt. Darauf folgt das Einerkomplement (logisch NOT) der Blocknummer und anschließend die 128 Datenbyte. Das abschließende Byte enthält die Prüfsumme der 128 Byte. Andere Implementierungen des Xmodem-Protokolls verwenden eine kompliziertere 2-Byte-Prüfsumme, die ein CRC-Check der 128 Byte darstellt.

Wird der Empfänger zuerst angestoßen, sendet er ein NAK (ASCII 15H), welches der Sender zur Synchronisierung der Xmodem-Prüfsumme verwendet. Wird das Zeichen »C« statt des NAK gesendet, dann wird die Xmodem-CRC-Prüfsummenberechnung verwendet. In jedem Falle sendet der Sender den ersten Block der Datei, den Block Nummer 1, wenn er das NAK oder das »C« erhält.

Nach dem Senden jedes Blocks führt der Empfänger die gleiche Kalkulation durch und vergleicht sein Ergebnis mit dem des Blocks. Bei einer Abweichung liegt ein Übermittlungsfehler vor.

Die meisten Empfänger ignorieren alle ankommenden Zeichen bis ein SOH empfangen wird, so daß der SOH-Teil die Prüfung ist, wodurch das Leitungsrauschen ignoriert wird. Die Blocknummer wird mit deren Komplement verglichen, um sicherzugehen, daß der Block in Ordnung ist. Ist es der erwartete Block, wird das ACK gesendet (ASCII 06H), und die Daten werden bearbeitet, indem sie normalerweise in einem Puffer abgelegt werden, der dann auf die Diskette geschrieben wird. Da viele Diskettenkontroller den Interrupt sperren, wird die Schreiboperation oft vor dem Senden des ACK durchgeführt, damit keine Zeichen verlorengehen.

Wenn die Blocknummer angibt, daß dies der letzte richtig erhaltene Block ist, so wird auch ein ACK gesendet, aber der Block nicht bearbeitet. Dies ereignet sich, wenn der Sender nie ein ACK für den angezeigten Block erhielt.

Ein NAK wird gesendet, wenn kein ganzer Block innerhalb der Timeout-Periode empfangen wird, das nächste Zeichen nicht innerhalb einer kürzeren Timeout-Periode ankommt oder einer der oben angegeben Prüfungen ergibt, daß ein Fehler vorliegt.

Wenn der Sender ein NAK erhält, zählt er einen Zähler hoch, der beim Erhalt eines ACK zurückgesetzt wird. Erreicht der Zähler einen gewissen Wert, so schickt der Sender ein CAN-Zeichen (ASCII 18H) und bricht die Übertragung ab. In einem Protokoll, das nicht stabil ist, bricht der Empfänger unmittelbar ab und ignoriert alle erhaltenen Daten. In einem robusten Protokoll wird das CAN mit NAK beantwortet, der Sender schickt das CAN nochmals, und beide Seiten brechen den Transfer ab. Ist der NAK-Zähler noch nicht auf dem Höchststand, wird einfach der letzte Block nochmal gesendet. Wird weder ein NAK noch ein ACK erhalten, tritt möglicherweise ein Timeout ein, welcher wie ein NAK behandelt wird.

Entweder wird die Übertragung abgebrochen, oder es wird ein Dateiendekennzeichen (EOF) gesendet. Wird das EOF erhalten, wird ein Ende der Übertragung (EOT, ASCII 04H) gesendet. Wird eine nicht robuste Version des Xmodem-Protokolls verwendet, wird sofort abgebrochen. In einer robusten Version wird NAK erwidert, das EOT nochmals gesendet und abgebrochen.

Xmodem ist aus mehreren Gründen das beliebteste Protokoll in der Kommunikation und der Dateiübertragung im PC-Bereich: Es wurde früh entwickelt (am Ende der 70er Jahre). Es ist einfach zu implementieren, es ist effizient, und das wichtigste: es funktioniert.

Alles über die „Internas“ der Hardware.





Alles über das Angebot an Peripherie und Zubehör, das „compaqtibel“ ist.

Aus der riesigen Software-Bibliothek für Compaq. Informationen und Empfehlungen zu Auswahl und Einsatz.

Die „Schnittstelle“ zum Anwender.

Für das Compaq-Magazin gibt es nur ein Thema: Compaq. Aber das ist ein Thema mit vielen Variationen. Und ein Thema mit vielen Seiten. Compaq-Computer sind mit die meist gekauften professionellen Personal Computer, und der Anwender-Kreis in Wirtschaft, Industrie und Wissenschaft wird größer und größer.

Erfahrungen, Problemlösungen, Anwendungen, Software und Know-How sind vorhanden. Ein riesiger Markt für Software, Peripherie und Zubehör ist entstanden. Neue Ideen, Techniken und Möglichkeiten bieten sich an. Für das Compaq-Magazin sind dies die Themen, um die sich alles dreht. Themen, die den Anwender interessieren, die ihm nützen und ihn mit Informationen versorgen für die Arbeit mit seinem Compaq.

Das Compaq-Magazin ist die „Schnittstelle“ zum Anwender.

COMPAQ MAGAZIN
November '87
Das Magazin für den Anwender
Die Neuen: Deskpro 386/20 Portable 386
Utilities: Die Festplatte optimal nutzen
Windows 386

Eine Zeitschrift aus dem IWT-Verlag.

NEU!

Bestellcoupon

Hiermit bestelle/n ich/wir:

- ☐ das Compaq-Magazin im Abonnement (vorerst 4 Ausgaben pro Jahr) zum Abopreis von DM 40,– (4 x 7,50 zzgl. Versandkosten). Ich kann das Abonnement 8 Wochen vor Ende des Bezugszeitraumes kündigen.
- ☐ ein Einzelheft zum Preis von DM 11,– (DM 8,50 + DM 2,50 Versandkosten)
- ☐ gegen Vorauszahlung auf unser Post-Girokonto München, Kto.-Nr. 99069-804, BLZ 700 100 80
- ☐ gegen Nachnahme ☐ gegen Rechnung

Meine/unsere Anschrift:

Name

Firma

Straße

Ort

Telefon

Datum, Unterschrift

IWT Magazin Verlags GmbH
Wendelsteinstr. 3, 8011 Vaterstetten

Faszination Programmieren

Satire, Lesestoff, Computergeschichten, künstliche Intelligenz

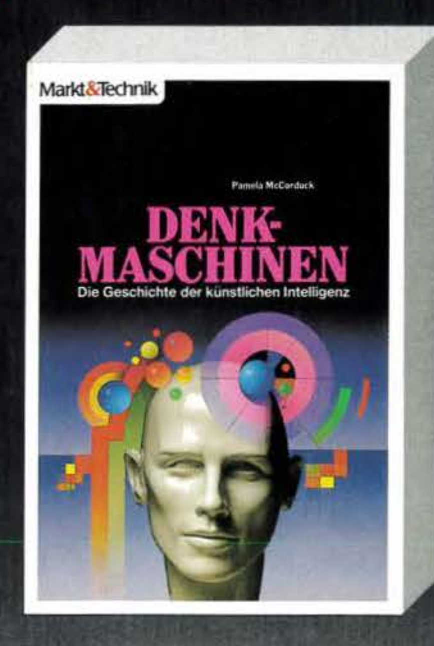
S. Lammers
Faszination Programmieren
1987, 430 Seiten

»Faszination Programmieren« ist eine Sammlung aufschlußreicher Interviews mit neunzehn namhaften Programmierern unserer Zeit. Die Interviews bieten einen Einblick in die Karriere jedes einzelnen, beinhalten Programmskizzen und Quellcode-Beispiele, beleuchten die Motivation und die Programmierbedingungen und schildern nebenbei die Geschichte der Mikrocomputerindustrie. Sie erfahren dabei, wie erfolgreiche Programmierer an ihre Aufgabe herangehen, ob Programmieren eine intuitive oder erlernte Fähigkeit ist, welche Entwicklungsmethodik Programmen wie VisiCalc, Microsoft Basic oder Lotus den Erfolg gebracht haben, die Hintergründe der Entstehung von Firmen und Forschungslabors wie Lotus, Apple, Xerox, PARC und Microsoft und vieles mehr.

- Ein packendes Buch – unerläßlich für den Fachmann und fesselnd für den Laien.

Bestell-Nr. 90418
ISBN 3-89090-418-1
DM49,–/sFr 45,10/öS 382,20





P. McCorduck
Denkmaschinen
1987, 344 Seiten

»Elektronengehirn«, so lautet eine alte, volkstümliche Bezeichnung für den Computer. Elektronengehirne waren noch raumfüllende Maschinen, und ihre »Intelligenz« hielt sich in engeren Grenzen als die eines heutigen Heimcomputers. Trotzdem haben die Computer vom Anfang ihrer Entwicklung an den Ruf gehabt, intelligent zu sein.
Dieses Buch erzählt die Geschichte dieser »Denkmaschinen«, die Geschichte der »künstlichen Intelligenz«. Die Autorin, Journalistin und Schriftstellerin, hat viele der wichtigsten Persönlichkeiten dieser Geschichte interviewt. Sie erzählt so auch die Geschichte der Menschen, die an dieser Entwicklung Anteil hatten. Das sind vor allem Ada Lovelace und Charles Babbage, Alan Turing, John von Neumann, Minsky, Newell, Simon und viele andere.

- Ein Buch, das sich an den Leser wendet, der mehr über die KI wissen will als Programmstrukturen und Sprachnormen.

Bestell-Nr. 90419
ISBN 3-89090-419-X
DM49,–/sFr 45,10/öS 382,20

K. Kupfernagel
Computer – die leisen Eroberer
1987, 269 Seiten

Was wissen Sie eigentlich über Computer? Kennen Sie seine Möglichkeiten und seine Funktionen? Was ist eigentlich eine Datenbank? Und, und, und. Wenn Sie sich diese und noch mehr Fragen stellen, dann ist das Buch »Computer – die leisen Eroberer« genau die richtige Informationsquelle für Sie. Es gibt Ihnen einen generellen Überblick zum Thema Computer und erläutert seine Fähigkeiten und Funktionsweisen. Ein Computer funktioniert nur nach wenigen Grundprinzipien – sie sind überraschend einfach! Wer diese Prinzipien verstanden hat, kann sich auch sinnvoll mit dem Computer auseinandersetzen. Mit einem Computer umzugehen ist heute wichtiger denn je; denn die Computerrevolution hat wahrscheinlich noch gar nicht richtig begonnen...

Bestell-Nr. 90179, ISBN 3-89090-179-4
DM 14,80/sFr 13,60/öS 115,44





E. Pawlu
Wenn der Computer Geschichten macht
1986, 161 Seiten

Millionen Zeitungsleser kennen Erich Pawlu, der die Entwicklungen in der Computertechnik mit Humor und Scharfblick verfolgt. Sein Buch »Wenn der Computer Geschichten macht« überträgt die Freuden und Probleme des elektronischen Zeitalters in amüsante Geschichten. Mit hintergründiger Satire verfolgt er die Veränderungen und die Möglichkeiten des Komischen, die durch die neuen Computer in unserer Umwelt und in unseren Köpfen entstehen. Einen besonderen Reiz bezieht dieser Band aus den nostalgischen Bildern, die mit »modernisierten« Bildtiteln einen zusätzlichen humorvollen Akzent setzen.

- Ein Buch für alle Computerfreunde und Computergegner die dem Computerzeitalter eine hintergründig-heitere Seite abgewinnen wollen.

Bestell-Nr. 90378, ISBN 3-89090-378-9
DM24,80/sFr 23,–/193,40)

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.



Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0.
SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656,
ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0,
Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526.



Fragen Sie bei Ihrem Buchhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

804166